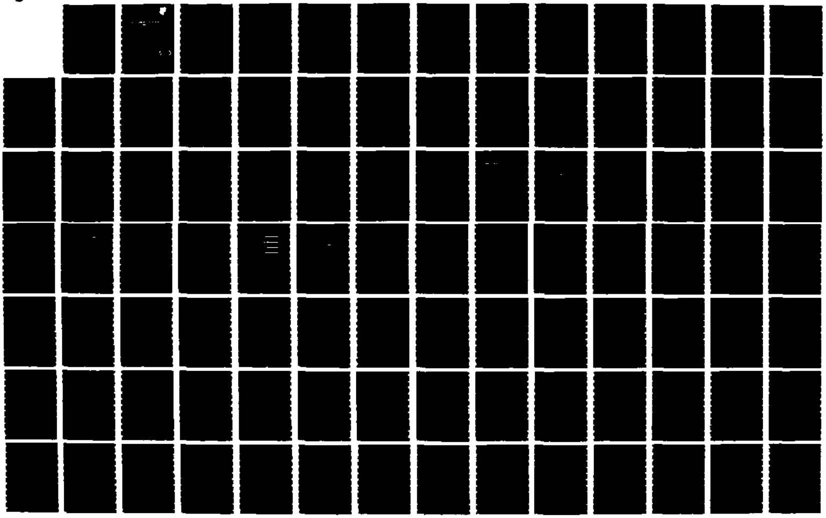
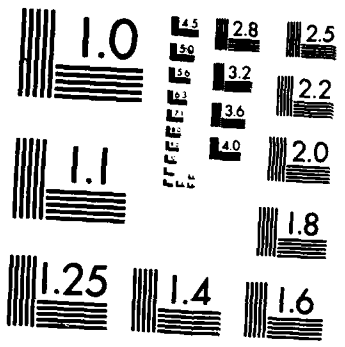
AD-A174 316   A PACKET COMMUNICATION NETWORK SYNTHESIS AND ANALYSIS   1/2
SYSTEM(U) STAR CORP LAWRENCE KS K S SHANMUGAN ET AL.
AUG 86 STAR-0112-1 RADC-TR-86-139 F30602-85-C-0112

UNCLASSIFIED   F/G 17/2   NL




1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
1.1
4.0
2.0
1.8
1.25
1.4
1.6

(12)

RADC-TR-86-139
Final Technical Report
August 1986

A PACKET COMMUNICATION NETWORK SYNTHESIS AND ANALYSIS SYSTEM

ST*AR Corporation

K. Sam Shanmugan, Victor S. Frost, G. J. Minden and E. Komp

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

RADC-TR-86-139 has been reviewed and is approved for publication.

APPROVED: [illegible]

DANIEL J. McAULIFFE
Project Engineer

APPROVED: Bruno Beek

BRUNO BEEK
Technical Director
Communications Division

FOR THE COMMANDER: John A. Ritz

JOHN A. RITZ
Plans & Programs Division

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ST*AR 0112-1	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-86-139
6a. NAME OF PERFORMING ORGANIZATION ST*AR Corporation 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (DCLF)
6c. ADDRESS (City, State, and ZIP Code) PO Box 3385 Lawrence KA 66046	7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) DCLF	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-85-C-0112
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62702F · PROJECT NO. 4519 · TASK NO. PR · WORK UNIT ACCESSION NO. OJ

11. TITLE (Include Security Classification)
A PACKET COMMUNICATION NETWORK SYNTHESIS AND ANALYSIS SYSTEM

12. PERSONAL AUTHOR(S)
K. Sam Shanmugan, Victor S. Frost, G.J. Minden, E. Komp

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Aug 85 TO Dec 85	August 1986	138

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
17	02	01	Packet Networks . Network Design Simulation Network Analysis

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

As communication networks become larger and more complicated, analytic modeling becomes increasingly more difficult and simulation plays an important role in performance evaluations. A key challenge is to be able to develop a flexible, computationally efficient and user friendly simulation package that can be used to analyze and design complex military communication networks that often operate in highly stressed environments. Under a PHASE-I SBIR contract, ST*AR Corporation has defined the requirements for a state-of-the-art network simulator, developed a prototype network simulator, demonstrated the feasibility of using the simulator for studying routing algorithms in a packet switched network, investigated computationally efficient techniques for network simulation, and identified several research issues that need to be addressed in the Phase II effort. Results of the Phase I efforts are described in this report.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Peter K. Leong	22b. TELEPHONE (Include Area Code) (315) 330-4567 · 22c. OFFICE SYMBOL RADC (DCLF)

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

1.1 Background

Performance evaluation and vulnerabilty analysis are the central issues in the design of military communication links and networks. It is extremely difficult to evaluate the performance of complex communication systems in closed form using analytical techniques alone except for some oversimplified and ideal cases. Digital simulation provides a useful and effective adjunct to direct analytical evaluation of communication system performance [1,2,3,4]. Indeed, there are many situations where explicit performance evaluation defies analysis and meaningful results can be obtained only through actual prototype system testing or digital simulations. The former approach is generally cumbersome, expensive, time consuming and relatively inflexible. These are considerations which typically weigh in favor of computer simulations.

Over the past decade, a large body of simulation based techniques have been developed for analyzing and designing communication systems at the LINK level [1]. At the network level, simulations have been limited in most cases to the analysis of queues [1], and a comprehensive CAD system is not available for networks even though several simulation tools and a few simulation packages are available. The tools that are available are programming languages (SLAM[5], RESQ[6], SIMSCRIPT[7], GPSS[8]), and they do not lend themselves to use by a network analyst with limited software background. It takes of the order of weeks to set up and verify network analysis using these tools.

Some network simulation packages have been developed in recents years (PANCEA,QNA,BEST/1,CADS,ASIM,GSIM [9]). Many of these packages were developed for specific applications and most of them are proprietary to the organizations which developed them. What is needed is a general purpose CAD system for communication networks that is available to a wide group of users including DoD research laboratories and contractors.

1.2 Objectives of the SBIR Network Simulator Program

The long term objective of the SBIR project is to develop a simulation based CAD package for networks that can be used to support the following applications:

Network Performance Analysis and Design
Network Vulnerabilty Analysis (structural as well functional)
Network Fault Identification and Reconstitution
Multi-Media Network Analysis and Design
Distributed Processing

The simulation software is to be used as the tool with appropriate interfaces to various applications in support of planning, analysis and design. The project has three phases: an initiation phase (I), a research and development phase (II), and a terminal phase (III).

The initial or Phase I objectives are the following:

1. Develop an approach and software design for network simulation.

2. Identify research areas to be addressed in Phase II and develop approaches to solving the research problems.

3. Develop a prototype network simulator and demonstrate the feasibility of using the simulator to study the effects of routing algorithms on network survivability.

Phase I effort started in July, 85 and has been completed in December, 85.

1.3 Results of Phase I Efforts

The performance of communication networks is greatly influenced by the topology of the network, protocols used, and the network management and control strategies employed. Any network simulation software that is developed has to be able to handle these network issues. While specific features of the network, say the protocol at ISO layer 3, may be handled through some of the currently available packages, no single package has the capabilities to address all network issues in an integrated environment.

In order to do this successfully, we propose to develop a new simulator under Phase II of this project. Our simulator will have a modular structure, utilize an interactive graphics user interface, use state-of-the-art hardware (workstation), appropriate programming language such as LISP, use proper statistical simulation techniques, and have built in AI/Expert System capabilities. Our simulator will use the graphics interface to specify the topology and structural parameters of the network. The graphics interface will also be used to provide static as well as dynamic display of results. Protocol aspects of the network will be handled by selecting primitive building blocks from a library and assembling the protocols for various layers. The dynamic performance of the network will be evaluated through an event driven simulation, and the network control and management function will be handled through an expert system.

A preliminary software development plan for the proposed simulator was completed during this phase and a prototype simulator was also developed and used to demonstrate the feasibility of the approach and the use of the simulator. Details are given in sections 2 and 3 of this report.

The prototype simulator that we have developed under the current phase represents an attempt to obtain a feeling for the structure and capabilities of a new generation communication network simulator. As such, the prototype is not a full fledged network simulator capable of detailed network analysis. The development of the prototype has, however, led to the further understanding of the structure and directions that a comprehensive network simulator should take.

Also in this current phase we have identified several research problems in simulation techniques. While brute force Monte-Carlo techniques can be used to simulate simple networks, such techniques will require several hours to several days of computer time to set up and simulate a complex network made up of a large number of nodes. Special simulation techniques have to be used in order to reduce the computational requirements. We have identified four approaches for improving the computational speed of network simulation. The first approach is based on a hybrid technique in which closed form analytical results are combined with simulation results. The second and third techniques employ special procedures for handling routine events such as background traffic in a network, and rare events such as the failure of links and nodes. The fourth approach will combine perturbation methods and simulation techniques. Details of these approaches, preliminary results, and problems to be solved in Phase II are presented in section 4 of this report.

The prototype simulator was used to investigate the effects of various routing algorithms. While this study met its goal of demonstrating the feasibility of simulation based analysis of communication networks, only simple routing algorithms were used in this preliminary study. While investigating routing algorithms that would be robust and guarantee network survivability under severely stressed operating conditions, we found that extremely sophisticated and self-adaptive routing algorithms are needed to assure network survivability. These algorithms are in general very complex, and can be implemented only under the framework of an expert system. This will be a major research and development task under Phase II.

Phase I results have clearly demonstrated the feasibility of our approach to developing an integrated CAD tool for networks. Our approach has several advantages over other contemporary approaches, and it is anticipated that the simulation based network CAD tool that we are developing will have wide applications in the planning, analysis and design of military as well as commercial networks.

2.0 A GENERAL PURPOSE NETWORK SIMULATOR: DESIGN APPROACH AND REQUIREMENTS

2.1 Introduction

Communication systems are composed of a hierarchy of functional components or blocks. At the lowest level are the electronic circuits constructed from either discrete or integrated components. The next level is made up of communication links which contain functional blocks like modulators, coders, filters, etc. At the highest level, there are communication networks made up of a variety of links and layers of network protocols and control (including software).

At the link level, existing CAD tools are uniformly implemented in four segments [1]. A system configurator segment allows the user to specify the topology of the link. In some systems, this phase uses extensive interactive graphics (ICS [1], Boss [10]), while in others the system configurator uses FORTRAN like statements. The functional blocks used in configuring the link are drawn from a library of subsystem models. For simulation based link CAD packages, typical library models are signal generators, encoders, modulators, noise sources, etc. The models are software routines which simulate the signal processing operations that take place in each functional block. Once the system is configured, a simulation exercisor performs the simulation of the system and stores waveforms at selected points in the link. Following the execution phase, a post processor is used to evaluate and view the results of the simulation.

Unfortunately, similar comprehensive CAD systems are not available for networks. Most of the packages that are currently available were written to simulate specific systems and do not have the modular structure that is needed for handling new applications and new systems. Also, many of the packages are designed either to run on a mainframe with the user input through a simple terminal, or on a PC. Both of these approaches preclude the use of state-of-the-art interactive graphics user interface with a quick response. An interactive graphics user interface with a good response time is essential if the CAD tool is to be productive. The general purpose network simulator that we are developing overcomes these shortcomings. Details of our approach are discussed below.

2.2 Overall Approach

While we can use an approach similar to the one used in link level CAD tools for developing simulation based network CAD tools, there are some differences in the nature of issues that are handled at the link and network levels which warrant slightly different approaches. The primary differences between network and link simulation are the following:

1. At the link level, one is simulating well defined hardware functions whereas at the network level we are simulating algorithms and software procedures (software simulation of software at the network level as opposed to software simulation of hardware at the link level).

2. Network simulation is event driven, whereas the link simulation is done in sampled time at the waveform level.

3. Topology plays a more important role at the network level.

4. Simulation outputs at the link level for the most part consist of time and spectral plots whereas at the network level the outputs consist of estimates of statistical parameters.

5. The computational requirements of a typical network simulation will be several orders of magnitude higher than the computational requirements of a typical link simulation. Hence, computational efficiency is an important problem in network simulation.

6. While simulators with fairly rigid structures (say for example fixed topology like in ICS [1]) are fairly useful at the link level, a great deal of flexibility needs to be built into network simulators.

In spite of these differences it is possible to use similar approaches for the overall structure of the link and network simulators. The major components of the network simulator should include a model library, a graphical structure, and a controller. The model library contains elements such as communication hosts, nodes, links, and building blocks necessary for implementing network protocols and management functions in a modular form. The graphical structure will be used to configure the network and to display the results of the simulation. The final portion of the simulator is the controller which is used to set up parameter values, to execute the simulation and to report the results. Details of these major components of the simulator are presented in section 3 of this report.

The simulator that we are developing is designed to run in a workstation environment and the programming is done in LISP. A workstation with a high resolution monitor provides a high quality graphics user interface and quick response. LISP type programming environment is ideal for developing software packages with a high degree of interaction and for implementing AI features.

2.3 Simulation Approach

In network simulation, we generate network traffic (messages or packets), route the traffic from the source to its destination and collect statistics on throughput, and delay. Occasionally we might want to analyze the effects of disturbances such as link and node failures on network throughput, delay and connectivity. A simple and brute force way of network simulation is to generate a large number of messages or packets and process them using a special program written to model the specific network being analyzed. While such an approach may be used for a one shot analysis of simple networks, a more flexible framework and computationally efficient procedures are needed for simulating complex networks.

Flexibility is needed to handle the wide variety of networks and protocols that are currently in use. By using a generalized ISO type layered model for the network, we break the network protocols used in each layer into a set of services provided, functions performed, and procedures used. We then develop state diagrams for the protocols used in each layer and define the interfaces between the protocols used in adjacent layers. While developing these models and interfaces, we identify the set of primitive building blocks that can be used to assemble a variety of protocols for each layer. Software procedures for these building blocks are developed and kept in a library that is used by the network configurator. With this modular approach we can, for example, change the media access protocols to simulate point to point, point to multipoint, or broadcast type networks without having to rewrite the entire simulation software. Thus we can handle complicated networks such as JTIDS, packet radio networks and LANs as well as simple packet switched networks, all in one integrated simulation environment.

While a modular structure provides flexibility, attention also has to be paid to the computational efficiency of execution of the simulations. Simulation based performance analysis can be carried out using either brute force Monte-Carlo techniques, or a combination of Monte-Carlo and analytical techniques. While the first approach provides maximum flexibility, it requires a large amount of computing resources. A combined approach reduces the computational requirements. However, it requires an expert analyst to set up and interpret the results. Thus, in a hybrid approach, we sacrifice some flexibility for improved computational efficiency and trade analyst's time for savings in computational requirements. We have identified several hybrid approaches that can be used to improve the computational efficiency of network simulation. Additional research will have to be completed in Phase II before these procedures can be implemented.

Details of the software structure and the simulation techniques are presented in sections 3 and 4 of this report.

2.4 Some Special Requirements

During Phase I research we have identified the following requirements as essential for the development and use of an efficient and useful network simulator.

1. Hardware requirement: In order to provide an effective graphics user interface, it is essential that the simulator be implemented on a state-of-the-art workstation with adequate processing power and graphics capabilities. The workstation should also be able to support high level programming languages such as LISP. We recommend the Xerox-1108, or a VAXSTATION II as possible candidates. It might be useful to have a highspeed network connection between the workstation and powerful host.

2. Software Environment: We recommend the use of LISP as the development language for the simulator. LISP offers several advantages over languages such as FORTRAN for developing software containing a high degree of interaction and for implementing AI/Expert System features.

3.0 A PROTOTYPE COMMUNICATION NETWORK SIMULATOR

3.1 Introduction

The present chapter describes a prototype communication network simulator developed under Phase I of the Small Business Innovative Research Contract. The prototype simulator represents an attempt to obtain a feeling for the structure and capabilities of a new generation communication network simulation system. As such, the prototype is not a full fledged network simulator capable of detailed network analysis. The development of the prototype has, however, led to further understanding of the structure of a comprehensive network simulator. In the present chapter, we first discuss a set of design goals for the prototype simulator. We then describe the programming environment and the structure for the simulator followed by a description of the results of implementing the prototype simulator and finally, a brief summary of the direction for future efforts in implementing a comprehensive communications network simulator.

3.2 Design Goals

The prototype simulator resulted from the consideration of four major design goals. These design goals include: 1) to establish a prototype communication network simulator structure; 2) to establish requirements for a protocol design and test tool; 3) to establish an interactive environment for network definition and display of results; and 4) to demonstrate the capabilities of a limited network simulator and interactive environment for packet switched networks. We expected the implementation of a prototype simulator to give us some indication of the structure of a comprehensive simulator. The resulting implementation did in fact give us some indication of this structure, and these are reported in the section on results.

Our goal of establishing a prototype for a protocol design and test tool was not quite as successful. The protocols implemented in the prototype simulator are directly implemented in the procedures for communication links and communication switches. This leaves little flexibility for the designer to test and verify alternative protocols for a network. While a protocol design tool was not implemented in the initial prototype simulator, we feel there are a number of possible approaches which will address this problem. These approaches are discussed in the section on future extensions.

The interactive environment and visual display capabilities of the prototype simulator perhaps demonstrate best the advantages of using a workstation approach to problem definition and analysis. Networks that would take several days or even weeks to configure using a conventional simulation language such as GPSS, SIMSCRIPT and SLAM are configured within minutes using the prototype simulator. Simulations of these networks can be run immediately once the network has been defined and as the simulation is running a visual display of the queue length and other information is presented to the designer. The designer thus obtains immediate feedback on the operation and performance of the network under study.

The final design goal was to demonstrate a simple network simulation for a packet switched network. We implemented a simple communications switch for routing packets through the network, a communications link for connecting switches together and a host for packet generation and end-to-end traffic. The demonstration network allows one to configure a packet switched network using communication hosts as sources and sinks of packets, communication nodes as switching points for routing decisions and communication links for connecting switches and hosts together. Parameters for all of the hosts, switches, and links can be set interactively and the results of the simulation are available on a visual display. The only limitation on the number of hosts, nodes and links is the size of the workstation the simulation is run on.

In the following sections, we describe the structure of the prototype simulator, the interactive environment and the packet switch network demonstration.

3.3 Prototype Simulator Software Structure

In the following sections we describe the software structure for the prototype communication network simulator. We will first describe the programming environment that we chose to implement the simulator in. We then describe the model graphics and control paradigm used to separate portions or areas of the simulator and then describe each of these portions in some details. The final section discusses the user interaction that is the network definition and simulation portions of the simulator.

3.3.1 Programming Environment

The prototype simulator was implemented in a combination of the Inter-Lisp-D and Loops programming environment on a Xerox 1108 workstation. The InterLisp-D programming environment provides a comprehensive set of tools for multiple display windows on the screen, tools for quickly inspecting data structures and modifying those data structures and tools for tracking changes to the programs as they evolve over time. The Loops object oriented environment provides a clean methodology for defining object structures (similar to records in PASCAL or MODULA) and the methods (procedures) that operate on those structures. Loops organizes the object structure and the methods which operate on that structure into an entity called a class. Classes exist in a class hierarchy in the Loops system. In this hierarchy, classes at a lower level called subclasses, inherit object structure and methods from classes at a higher level in the hierarchy. Classes may exhibit a single inheritance chain from a low level class to higher level classes in which case the low level classes are special versions of the higher level classes or low level classes may have multiple inheritances from higher level classes in which case the low level classes exhibit a broader range of behaviors from each of their super classes. For a further description of the capabilities of the loops programming system, the reader is directed to [11].

The Loops system on the Xerox 1108 provides an interactive and graphical means of defining new classes, their structure and their methods. Developing the prototype simulator then became a task of defining the necessary classes, the structure of each class and the methods to perform the simulation. These classes are described in the Section 3.3.3.

3.3.2 The Model-Graphic-Control Paradigm

The approach taken in developing the prototype simulator was to separate the functions of the simulator into three areas or portions. The first area we called the model which consists of the classes which are necessary to carry out the simulation of a communication network. These classes deal only with the communication aspect of the network and are not concerned with either the interactive portion of the simulator, nor controlling or configuring the network. Examples of classes in the model portion of the prototype simulator are CommHost, CommNode and CommLink. The second area of the simulator is graphic, which is concerned with graphics and interaction with the user. A graphical structure parallel to the communication structure was set up within the simulator. This graphical structure was used to configure the network and to display the results of the simulation. Within this graphic area are classes for GraphHost, GraphNode and GraphEdge. The final area of the model-graphic-control paradigm is <u>control</u>. The control area is used to define a communication network, to set the parameters for that network and to execute a simulation and to report the results of that simulation. The control area of the simulator consists of classes for packets, a packet handler, a task master, a graphic view, and a network operator.

We found that while it is quite useful to separate the functions of the model, that is, the communication portion of the simulator from the graphical and interactive portion of the simulator, there was a certain amount of redundant work in building the parallel structures. In future designs, the communication and graphical classes will still exist as separate classes but will be combined into a single class which will represent both the communication and graphical aspects of a communication entity.

3.3.3 Communication Classes

The prototype simulator implements four communication classes. These are CommDevice, CommLink, CommNode and CommHost. The CommDevice class represents that structure and behavior that is common to CommLinks, CommNodes, and CommHosts. The CommLink class represents a point-to-point communication link between either two CommNodes or a CommHost and a CommNode. The CommNode class represents a communication switch or routing mechanism within a packet switched network. The primary task of the CommNode is to implement the routing strategy for the network. The final communication class is called a CommHost which is a specialization of a CommNode. The CommHost has the additional capability of generating packets for the network and terminating packets in the network.

3.3.3.1 CommDevice

The CommDevice class implements the structure and methods that are common to both CommLinks, CommNodes and CommHosts. This structure includes a queue of packets waiting for service in each of the communication objects or elements, e.g., links, nodes, hosts, etc., the length of that queue, the network that the communication object belongs to, a collection of statistics and the status of the communication object. The CommDevice class contains methods for changing the status of the communication object. The status may change from available to unavailable or from unavailable back to available. This change in status represents links or nodes going up or down as the simulation progresses. The important aspect concerning a change in status method is that if a CommLink goes down, the packets which are present on the CommLink's queue are removed from that queue and placed on the CommLink's source switch's queue for rerouting. Thus, packets that have been routed once and assigned to an outgoing link and then having the link go down are rerouted through alternative routing if available. The CommDevice class contains methods for placing a packet into a queue for a communication device, removing a packet from a queue and clearing the queue setting it to zero. The CommDevice also contains methods for gathering statistics about packets which are serviced by each CommDevice, initializing those statistics at the beginning of a simulation and to compute resulting statistics from those gathered during the simulation. The CommDevice implements the methods for terminating a packet and handling packets which are undeliverable due to insufficient routing information or unavailable CommLinks.

3.3.3.2 The CommLink

The CommLink class has a queue and queue length which represent the packets waiting to be transmitted on that CommLink. Each CommLink maintains its own set of statistics on the number of packets which have been transmitted on that link and delay information about the traffic on that link. Each CommLink is assigned a source node and a destination node which are CommNodes or CommHosts. Each CommLink has a capacity in bits per second, a bit error rate and a physical length. The capacity is used to determine the transmission time over the link. In the prototype simulation the bit error rate and the physical length fields were not used. The primary methods for a CommLink are to initialize the statistics and the link for simulation, to service a packet, and for network definition to make the link a full duplex link. The most important method is packet service on the CommLink. To do this, the CommLink first determines its current status, that is, whether the link is available for transmission or whether the link is down and unavailable for transmission. If the link is available for transmission the queue is checked for a packet. If a packet is found, that packet is transmitted on the link and an event is scheduled for termination of that transmission. When the transmission has terminated, statistics are gathered for that packet and the packet is queued on the destination CommNode. The queue is rechecked for additional packets and if one is present a new packet is transmitted. If no packet is present in the queue, an event to check the queue again in the near future is scheduled.

Note that each CommLink includes its own queue and queue handling facilities. While in a conventional network the queue actually resides in the communication switch, in the simulation the queue is incorporated into each communication link. The addition of links to a switch simply increases the number of queues available for the switch and no limitation is placed on the number of links which can attach to a switch.

3.3.3.3 The CommNode

The CommNode class has a similar queue structure and statistics to the CommLink class. That is, it includes a queue, a queue length, statistics field and a status. The CommNode, however, includes a capacity in terms of packets per second which the CommNode can handle, a list of the inbound links, a list of the outbound links and routing information. The CommNode capacity is used to establish the number of packets per second that the CommNode can handle in the network. The inbound links and outbound links provide a list of inbound connections and outbound connections and the routing information field provides information on how to route packets to each possible host in the network. The CommNode is slightly more complicated than the CommLink in that it must perform a routing function and must also be able to terminate packets if they have reached their destination or cannot be routed. The CommNode also implements a simulation initialization method to initialize the statistics and status of the CommNode.

The most important method within a CommNode is the service packet method that controls the service of packets. To service a packet, the CommNode first checks its status and if the CommNode is available, it then attempts to obtain a packet from its input queue. If a packet is available, statistics on the packet are gathered as to the amount of time it has spent in the queue. The packet is checked for a maximum hop count and the hop count is incremented by one. If the packet has reached its destination, the packet is terminated and no further action takes place. Otherwise, the packet is routed to its next switch using the current routing strategy. Once routed, the CommNode schedules an event to service another packet in the near future based on its capacity in number of packets per second.

To route a packet the CommNode uses one of two global strategies. The simplest strategy is a flooding strategy in which each packet is copied and the copy is sent on all outgoing links. The hop count field of a packet is used to set a maximum limit on the number of hops a packet can take before being terminated within the network. The second strategy is a fixed routing with alternatives approach. In this strategy the CommNode examines the destination field of the packet and uses a list of outgoing links for that destination to route the packet. The first link on this list that is available is used to transmit the packet. The packet is then queued on that CommLink's queue and it is then up to the CommLink to carry out the actual transmission of the packet. Since the list of outgoing links for each destination may contain one or more outgoing links this represents a simple but somewhat robust scheme for routing packets around links and/or nodes that are unavailable during the simulation. The user specifies the primary and secondary routes from each switch to each destination as part of the network configuration step.

While only simple routing schemes are implemented in the prototype simulator, "expert" routing algorithms will be implemented in the next version. An expert systems framework for implementing adaptive routing algorithms will be developed and incorporated in the next phase.

3.3.3.4 CommHost

The final communication class is a CommHost. The CommHost implements all of the capabilities of the CommNode plus the capabilities to generate traffic from a CommHost to all other CommHosts in the network. The actual traffic between CommHosts is obtained from a traffic matrix that the user defines during network definition. The CommHost class contains a method to initialize the CommHost for simulation which includes a step of determining all of the other hosts that are available in the network. For every other host that exists in the network, a generate packet event is scheduled in the future based on an exponential interarrival time whose mean value is based on the traffic between the source host and the destination host. The generated packets are placed on a queue local to the CommHost and packets on this queue are serviced in a manner identical to those on a CommNode.

3.3.4 Graphic Classes

There are four graphic classes, one corresponding to each of the communication model classes. The basic class is called a GraphObject and contains that portion of the structure and methods that are common to the other three graphic classes. The GraphEdge class is used to represent CommLinks on the display screen. The GraphNode class is used to represent communication switches and the GraphHost class is used to represent communication hosts. The primary difference between the GraphNode and GraphHost classes are the shapes that are drawn. The GraphNode class draws a circle and the GraphHost class draws a hexagon.

A graphic structure parallel to the communication structure is set up when the network is defined so that for every CommLink in the communication network there is a GraphEdge representing that CommLink in the graphical structure. For every CommNode in the communication network, there is a GraphNode in the graphical structure and for every CommHost in the communication network, there is a GraphHost in the graphical structure.

3.3.4.1 GraphObject

The class GraphObject contains that structure which is common to all graphic objects. This includes an attribute list, the represented object, the window the graphic object is displayed in, an x location, and a y location within that window.

GraphObjects know how to draw themselves in several ways including the initial draw, a temporary draw mode and to erase themselves from the display window. GraphObjects are also capable of displaying information provided to them either by a vertical bar above the GraphObject or by having numbers attached to the GraphObject. Illustrations of this capability are provided in the section on user interaction. The final capability of a GraphObject is to test whether it is near to a sample point in the window. This capability is used by the user interface to allow the user to select a GraphObject (or its corresponding communication object) by pointing at the object on the display.

3.3.4.2 GraphEdge

The GraphEdge class is used to represent a CommEdge in the communication network. In addition to the standard GraphObject structure, a GraphEdge has a destination node and a source node field to indicate the GraphNodes that it is connected to. The GraphEdge knows how to draw itself, a curved line between the destination node and the source node, how to calculate the trajectory of that curve, how to delete itself from the drawing.

3.3.4.3 GraphNode

The GraphNode class includes the structure of the GraphObject class plus a list of inbound edges connected to it and a list of outbound edges eminating from it. The class knows how to draw a node, which is a circle, how to delete the node from the drawing, how to move the node about on the drawing, and how to detach an edge from itself.

3.3.4.4 GraphHost

The GraphHost class inherits all of the structure of the GraphNode class and has a specific method for drawing a GraphHost which is to draw a hexagon rather than a circle.

3.3.5 Control Classes

The control portion of the simulator consists of classes for packets, a packet handler, a task master, a graphic view and a network operator. These are the Packet, the PacketHandler, the TaskMaster, the GraphView and the NetOperator respectively. The PacketHandler class provides a means for allocating and deallocating packets used by the simulation. The TaskMaster class contains the structure and methods for keeping track of tasks (events) to perform in the future, scheduling those tasks, and executing those tasks. The GraphView task provides a structure and method for creating a display window on the screen, providing pop-up menus for the left mouse button, middle mouse button, and title bar. The NetOperator class incorporates the above listed structure and methods and adds additional structure and additional methods to create the user interface between the network designer and the simulation system. We shall first discuss each of the super classes that make up the NetOperator and then discuss the additional capabilities of the NetOperator.

3.3.5.1 PacketHandler and Packet Classes

There are two classes which deal with the packets used in the simulation system: the Packet class represents packets and the PacketHandler class which manages packets. The structure of a packet consists of the source communication node, the destination communication node, the creation time of the packet, the queue time of the packet on the most recent queue, the number of bits in the packet, the hop count, a priority (which is not used in the current simulation) and the containing message or information bits. There are no methods associated with the Packet class.

The PacketHandler class is used to maintain a queue of free packets for use in the simulation. This class is implemented primarily to save on garbage collection and reduce the amount of memory needed for the simulation. The PacketHandler class contains a single field called a packet queue. The PacketHandler class contains three methods for allocating packets, deallocating packets and copying packets. The allocate method first checks the packet queue for a free packet and if one exists, removes it from the queue and returns it as a result of the allocation. If no packet exists on the queue, a new packet is created and returned. The deallocate packet method returns the given packet to the packet queue for use at a later time. The copy packet method makes a copy of the given packet and returns the copy to the caller.

3.3.5.2 TaskMaster Class

The TaskMaster class implements a capability for storing a list of tasks to execute some time in the future, methods for scheduling and executing those tasks. The TaskMaster structure includes fields for a start time and a stop time for the simulation, and the current simulation time which is called task time. There is a task queue which is a list of tasks to be done in the future and two flags, a pause flag for pausing the simulation, and a task verbose flag for printing out information as the simulation takes place.

The TaskMaster implements a number of methods for implementing tasks. There is a method to initialize the TaskMaster for a simulation. There are two methods for scheduling tasks, one to schedule a task relative to the current time and a second method to schedule a task at an absolute time. There are two methods for executing tasks, one to execute a single task and return to the caller for single stepping through a simulation, and one to execute all tasks on the task queue until the stop time of the simulation or the pause flag is set. There are methods for determining all of the tasks scheduled for a CommDevice and to delete specific tasks from the task queue. These methods are used when a link or node goes down to remove its tasks from the task queue. Finally, there are methods to set the pause and verbose flags within the TaskMaster.

3.3.5.3 GraphView Class

The GraphView class implements a structure and methods to support the user interaction with the simulator. The GraphView class is a subclass of the Loops Window class which allows one to create and position a display window on the screen and to attach menus to each of the mouse buttons (see Section 3.4 for description of mouse operation) and the title bar of the window. The GraphView class contains fields for a list of the objects within the graphic window, the width of lines drawn in the window, x, and y scales. In addition, the GraphView class itself contains a field for the menu items which will appear when a left button event occurs in the title bar of the window and when the middle button occurs within the window. These menus are discussed in the section on user interface.

The GraphView class implements a number of methods for manipulating graphic objects within the display window. These include methods to add GraphNodes, GraphEdges and GraphHosts to the display and to delete edges, nodes and the entire graph from the display. The GraphView class also implements methods for removing a graphic object from the list of known objects, to make an edge giving a source and destination node, and to move a node about on the window. The GraphView class also contains a number of methods for facilitating user interaction and debugging the system. These methods include the capability to inspect the NetOperator, to inspect the GraphObjects making up the display, and to inspect the communication objects making up the communication network. The user can select these functions from the title bar menu and then simply point at the object on the display he wishes to inspect and a new window appears which shows the field names and contents of those fields for the objects selected. The GraphView class contains the methods for finding which GraphObject the user is pointing to within the window and return that object. The GraphView class also handles scaling of the drawing within the window so that larger windows can be used for easier viewing by the designer.

3.3.5.4 The NetOperator Class

The NetOperator class combines the capabilities of the PacketHandler, TaskMaster and GraphView classes and adds additional structure and methods for defining the communication network and running the simulation of that network. The NetOperator adds the following fields: 1) a list of all the communication network objects in the simulation; 2) four matrices for controlling traffic and gathering statistics about host to host traffic in the network; 3) a list of disturbances to the network for use during the simulation; and 4) a verbose flag used to print additional detailed information concerning the simulation. The matrices are the Traffic Matrix view, the Generated Traffic view, the Terminated Traffic view and the Network Delay view. The Traffic Matrix view defines the traffic between hosts in the network. The Generated Traffic view records the actual traffic generated between hosts in the network. The Terminated Traffic view records the traffic that has actually reached its destination in the network and the Network Delay view records the total delay time between hosts in the network. Each of these view is a separate window on the display screen and can be manipulated by the designer in defining the network simulatiou. Disturbances consist of an absolute time, a disturbance type and the communication objects to apply the disturbance to. During simulation initialization the NetOperator schedules each of the disturbances listed as tasks for the TaskMaster to execute at the specified time. The NetOperator class implements a menu for the left mouse button when a left button event occurs within the display menu. The menu which pops up allows the designer to control the simulation and the result display for the simulation.

The NetOperator class implements a large number of methods for user interaction with the simulation. These include: 1) methods for adding communication links, hosts and nodes to the communication network; 2) methods for defining the routing strategy and the actual routing paths within the network; 3) methods for defining and initializing disturbances within the network; 4) methods for initializing the simulation, to start a simulation, to pause a simulation, and to continue a simulation from a pause. Once a simulation has been completed or paused, the designer can display the results of that simulation using either a bar chart display or by listing numbers next to each communication object in the display.

3.4 User Interaction

In this section, we describe the interaction between the system and the user in defining and executing a simulation. The primary means of interaction is a high resolution monochrome display capable of displaying a large number of windows simultaneously. The user directs the system via a mouse and a keyboard with the primary interaction device being the mouse. A typical screen is shown in Figure 3.1 which illustrates an executive window in the upper left hand corner, a partial network in the center of the screen, several windows showing traffic information, a window indicating the current simulation time and various other windows in the upper right hand corner that are part of the InterLisp-D system.

The primary mode of user interaction is via a two button mouse and pop-up menus within the network display window. Each mouse button (left and right) plus the combination of both mouse buttons (called middle) cause a specific mouse event to occur which in turn causes a menu to appear within the network window. The right hand mouse button causes a general purpose window manipulation menu to appear which enables the user to reposition the window on the screen, change its shape, shrink it to a small icon, etc. The network simulation is built around the left mouse button and the middle button. The middle button is used to define the network and interconnetions and the left button is used to define and initiate a network simulation. In addition, there is one other menu which appears if either the left or middle button is pressed when the cursor is in the title bar of the network window. This menu is used primarily for debugging the system and allows the user quick access to the data structures which define the network and the display.

3.4.1 Illustration of User Interaction

In this section, we present illustrations of the user interaction. A number of pictures and explanations are provided for major steps in the design and simulation of the network.

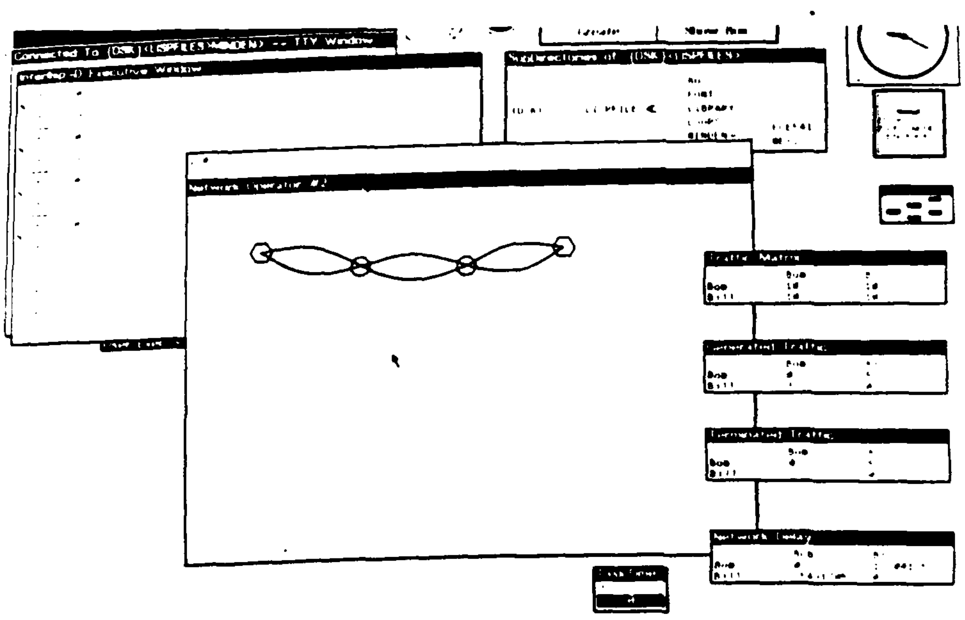


Figure 3.1

Figure 3.1 illustrates a typical display screen on the Xerox 1108 workstation. In the center of the screen is a fairly large window for displaying the communication network. In the upper left hand corner behind the network window is the user interaction window for typing commands (i.e., executive window). To the right of the screen are four windows displaying traffic information and in the bottom center of the screen is a small display indicating the simulation time. There are other miscellaneous windows in the upper right hand corner of the display.

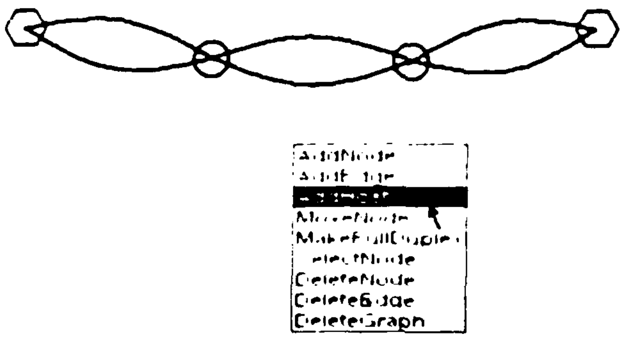


Figure 3.2

Figure 3.2 illustrates the network definition menu and a partially completed network. The user has positioned the mouse cursor to select an AddHost command from the menu. This menu appears when the user presses both the left and right mouse buttons simultaneously within the network display window.

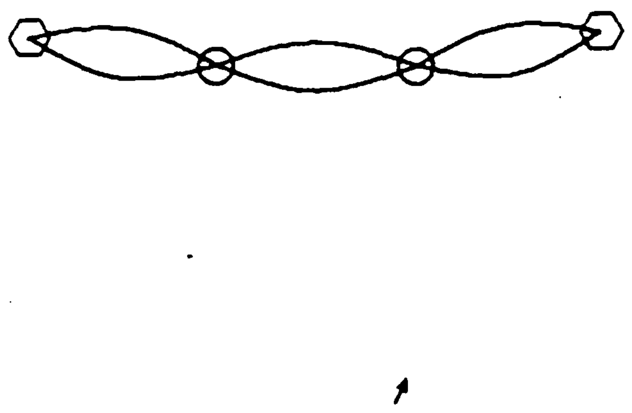


Figure 3.3

Figure 3.3. When asking for positional information within the network window, the cursor changes from an upward left pointing arrow to an upward right pointing arrow. This is an indication to the user to select a position within the network window.

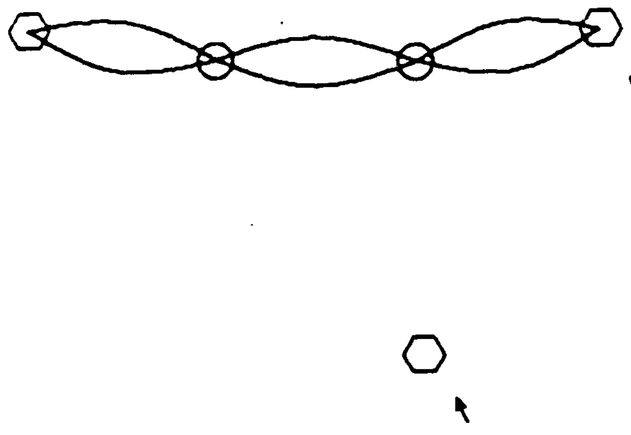


Figure 3.4

Figure 3.4. The user has positioned a new communication host, and that is drawn as a hexagon in the Figure.

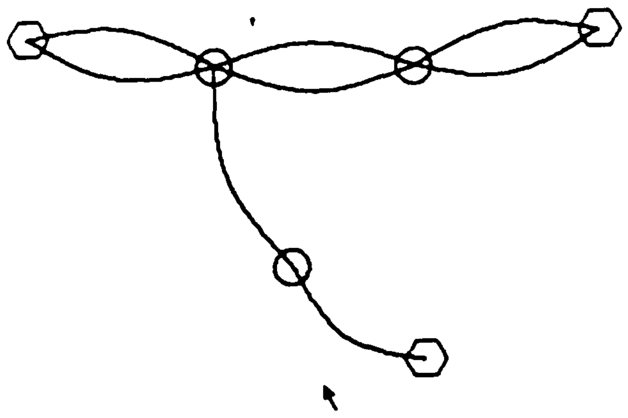


Figure 3.5

Figure 3.5. The user has selected the AddNode and the AddEdge commands from the network definition menu to continue building the communication network. CommLinks in the network are defined by pointing first to the source node or host and then to the destination node or host. When pointing to a node or host, the user must position the point of the arrow within the circle or hexagon. The network is completed by selecting the MakeFullDuplex item.

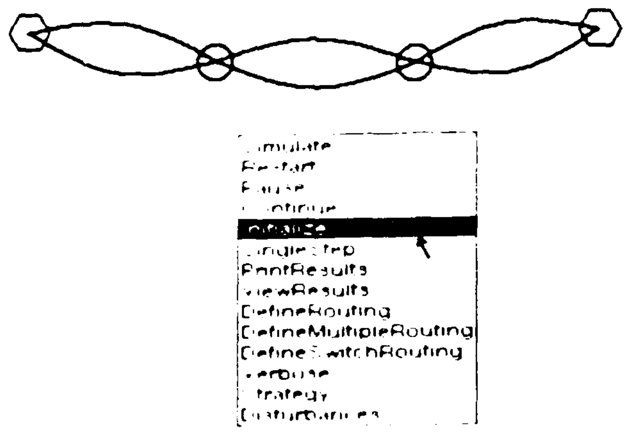


Figure 3.6

Figure 3.6 illustrates the simulation control menu. This menu appears in the network window when the user presses the left mouse button while the cursor is within the network window. The options in this window are described in the text of the report. In this illustration the user has selected the INITIALIZE item from the menu.

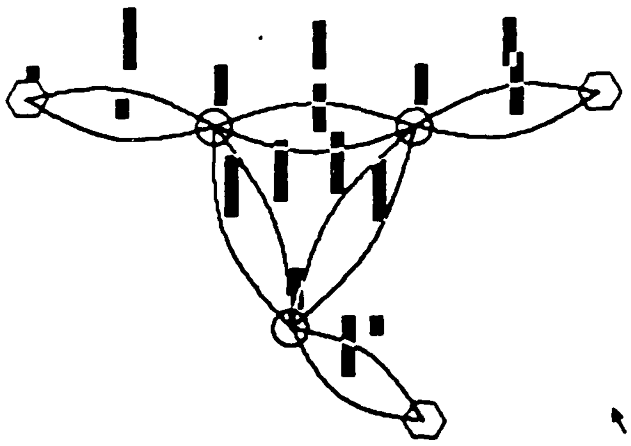


Figure 3.7

Figure 3.7. When a simulation is initialized and started, the window displays the queue length of packets on each host, node and link as a vertical bar near each entity. A typical display is shown above.

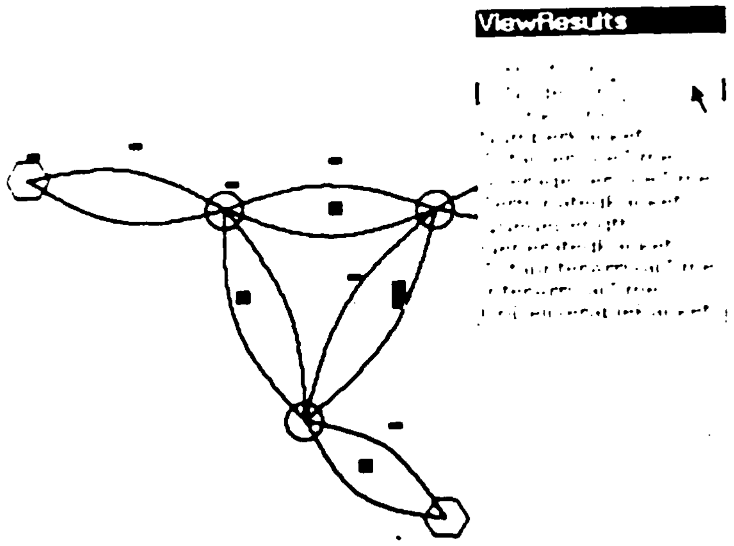


Figure 3.8

Figure 3.8. In addition to the queue length at each communication entity, the user can obtain other statistical information either on the display or via print outs. Figure 3.8 shows the pop up menu for selecting information to be printed out in a window.

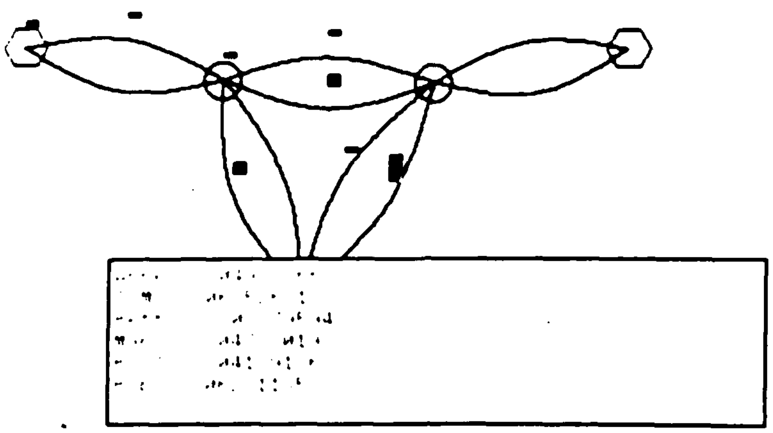


Figure 3.9

Figure 3.9 shows a typical print out for one of the statistics.

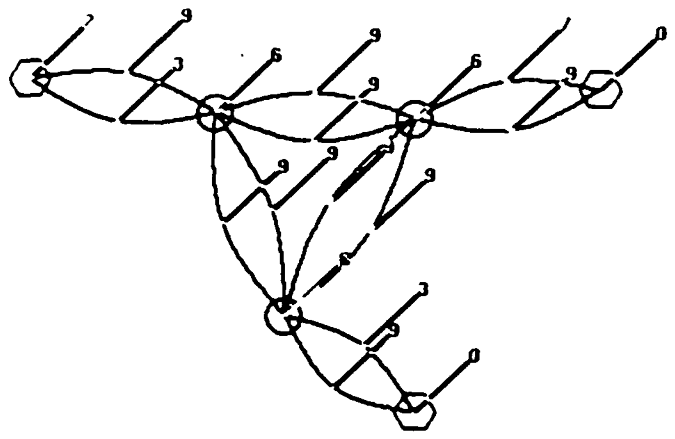
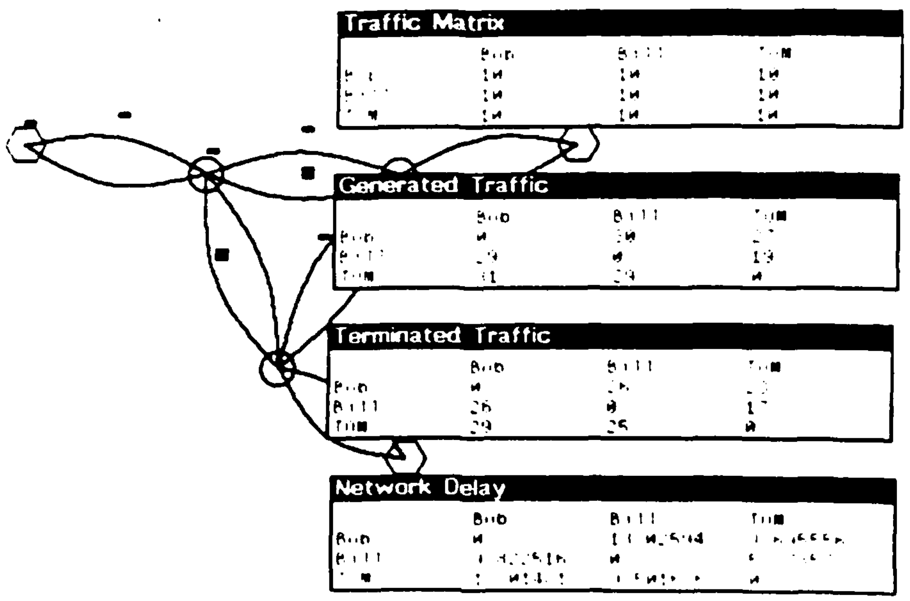


Figure 3.10

Figure 3.10 illustrates an alternative way display information on the network graph. Rather than use a vertical bar, a number is printed near each entity.



Traffic Matrix

	Bob	Bill	Tom
Bob	10	10	10
Bill	10	10	10
Tom	10	10	10

Generated Traffic

	Bob	Bill	Tom
Bob	0	30	[illegible]
Bill	29	0	13
Tom	31	29	0

Terminated Traffic

	Bob	Bill	Tom
Bob	0	26	[illegible]
Bill	26	0	1[illegible]
Tom	29	25	0

Network Delay

	Bob	Bill	Tom
Bob	0	[illegible]	[illegible]
Bill	[illegible]	0	[illegible]
Tom	[illegible]	[illegible]	0

Figure 3.11

Figure 3.11 illustrates four matrices used to record host to host traffic information. The Traffic matrix defines the traff'- rate between each pair of hosts in the network in messages per second. The Generated Traffic matrix records the actual amount of traffic generated during the simulation. The Terminated Traffic matrix records the actual number of packets terminated between each pair of hosts. The Network Delay matrix records the accumulated delay between each pair of hosts.

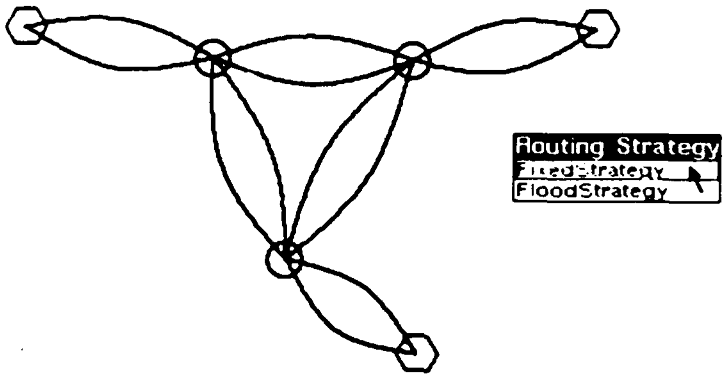


Figure 3.12

Figure 3.12 illustrates the menu for selecting the routing strategy. In this case the user has selected the fixed strategy which allows the user to define the routing from each switch to each possible destination.

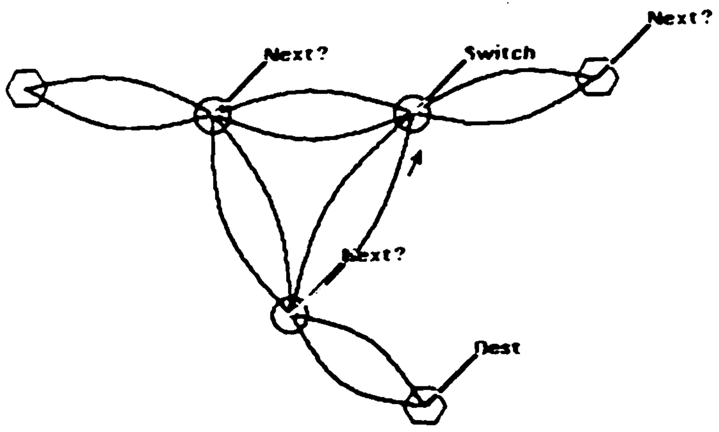


Figure 3.13

Figure 3.13 illustrates one step in defining the routing from a switch to a destination. The destination host is highlighted with the label "Dest." The switch for which one is defining the routing for is labeled with "Switch," and the possible next switches are indicated with "Next?". The user selects the next switch by pointing at the switch with the mouse cursor.

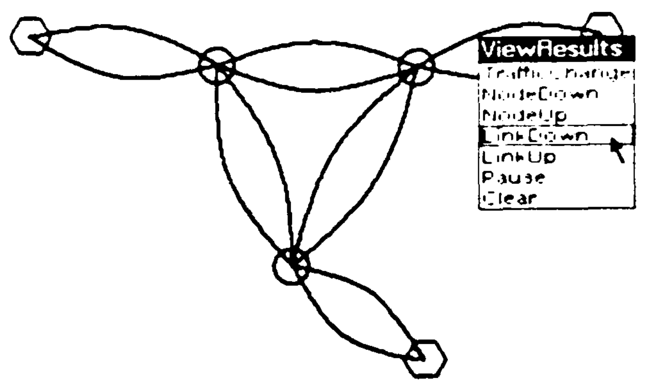


Figure 3.14

Figure 3.14 illustrates the selection of disturbances for the network. The user can program either a traffic change, a node or link going up or down, a simulation pause or clear disturbances into the simulation. In the illustration, the user has selected the link down disturbance.

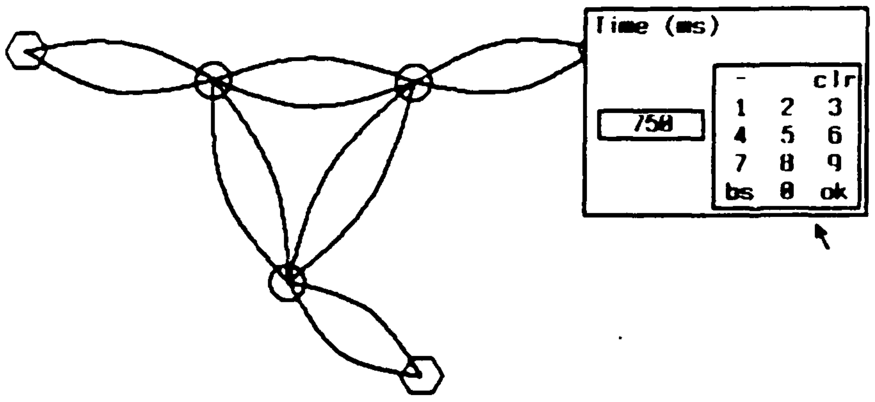


Figure 3.15

Figure 3.15 illustrates the specification of the disturbance time in milli-seconds. The user simply points at the digits in the menu and uses the left mouse button to select a digit. The user selects OK when complete. The user must then point to the link or node that is involved in the disturbance to complete the disturbance definition.

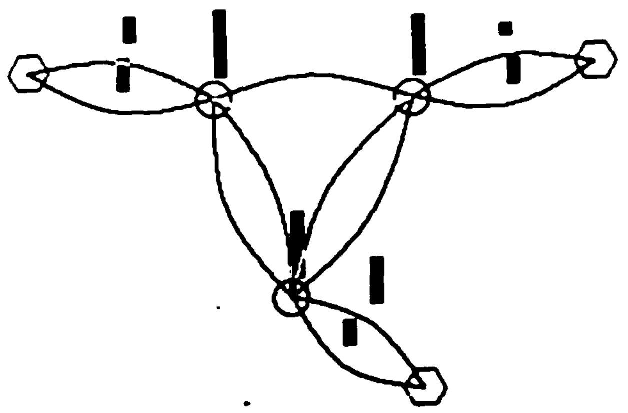


Figure 3.16

Figure 3.16 illustrates the display when a disturbance takes effect. Note that a communication link has been removed from the top center pair of links in the communication network. The packets that were queued for transmission on that link have been returned to the source node for rerouting.

3.4.2 Network Definition Menu

When the middle button is pressed within the network window, the network definition menu appears at the cursor position. The user has the option to select from the following items in that menu.

1. Add Node A new node is added to the network at the position indicated by the cursor.

2. Add Edge A new edge is added between existing nodes or existing hosts in the network. The user indicates the source node and then the destination node using the cursor.

3. Add Host A new host is added to the network. The host is indicated by a hexagon.

4. MoveNode The move node selection allows the user to reposition a node within the network. This is a limited capability to allow the user to make the network look better.

5. MakeFullDuplex insures that each connection in the network is bidirectional.

6. DeleteNode A node or host is deleted from the network.

7. DeleteEdge An edge is deleted from the network.

8. DeleteGraph All nodes, hosts and edges are deleted from the network.

The use of this menu is shown in Figures 3.2 through 3.4 which illustrate the menu itself on the network window and then the addition of a new node to the network.

In Figure 3.2 the user has simultaneously pushed the left and right mouse buttons to cause a middle button event and the network definition menu has appeared within the network window. The user has selected the item AddHost to add a new host to the network from that menu. The result is shown in Figure

3.3 where the direction of the cursor has changed indicating the system is waiting for a position for the new node and once a left button is pressed, the new host appears Figure 3.4. Both nodes and hosts can be added in this manner. To add a new edge to the network the user first selects the AddEdge item from the menu and then indicates the source node or host on the network by the cursor and then the destination node or host on the network.

To delete a node or host from the network the user selects the DeleteNode item and then using the mouse and cursor points to the node or host to be deleted. The arrow point of the cursor must be within the circle or hexagon of the node or host for a valid selection. To delete an edge from the network, the user selects the DeleteEdge item from the menu and then must point to the midpoint of the edge to indicate which edge is to be deleted.

When a new host or node is added to the network, the user must supply a name for that host or node. The system prompts for this name in a prompt window immediately above the network window. Note that in the prototype system it is important to provide this node or host name. However the prototype system does not provide an error message if a name is not provided. Also, when a new host is added to the networks, the host name is also added to the traffic matrices associated with the network. These are the Traffic Matrix, the Generated Traffic, the Terminated Traffic and the Network Delay matrices. These matrices all indicate host to host traffic and statistics.

3.4.2 Simulation Control Menu

The second major interaction menu in the system is used to control the simulation of the network. This menu has the following items and uses:

1. Simulate initiates a simulation of the network starting at simulation time zero.

2. Restart resets all the statistics gathered so far for the network and then continues the simulation.

3. Pause pauses the simulation so the user can observe results.

4. Continue continues the simulation from a pause state.

5. Initialize initializes all of the queues in the network to zero and schedules initial events for each communication entity in the network.

6. SingleStep allows the user to execute one event at a time, observing the results after each event execution.

7. PrintResults allows the user to print out a number of statistics about the simulation.

8. ViewResults allows the user to see on the network window a number of results of the simulation.

9. DefineRouting, DefineMultipleRouting and DefineSwitchRouting are used to define routing tables for the different routing algorithms implemented in the prototype.

10. Verbose causes a detailed print out of the events in the simulation

11. Strategy allows users to select one of two strategies for the routing algorithms in the prototype system.

12. Disturbances allows the user to program disturbances to the network that occur during the simulation.

Once a network has been defined, the designer must also: 1) select a routing strategy; 2) select a display format; 3) define a host-to-host traffic matrix, 4) define any disturbances; 5) initialize the simulation; and 6) start the simulation.

The first action is to select a strategy using the Strategy item from the menu. Currently two strategies are available, a fixed routing strategy with alternative routes, and a flooding algorithm. If the fixed strategy has been chosen, then the user must define the routing for that fixed strategy using the DefinedSwitchRouting item in the menu. The definition of the routing is shown later in this section. The second action is to define the method for displaying the results of the simulation while the simulation is running. To do this the user selects the ViewResults item from the menu and then selects either the PointOut or BarEntry in that menu. Third, the user must define the host-to-host traffic in messages per second. This is done using the Traffic Matrix and the left button menu which appears when inside of that window. Fourth, if any disturbances are to be programmed into the simulation these must be defined using the Disturbances item in the main menu. Fifth, the network must be initialized selecting the Initialize item in the main menu and finally, the simulation is started with the Simulation item in the menu. Note that in the prototype system all of these steps must be taken to initialize a simulation of the network. However, the prototype system does not check if these steps have been taken nor leads the user through these steps and so if a step is left out an error is possible. Such checking will be addressed in Phase II.

During the simulation of the communication network the queue length at each node, host and link in the network is shown as a vertical bar within the network window. The height of this bar indicates the number of packets which are waiting at that communication object for processing. Initially, when a simulation is started the user will see packets being generated at each of the CommHosts and the queue length for processing built up at each host. Some time into the simulation (currently defined at one-tenth of a second) each of the CommNodes, CommHosts and CommLinks will begin servicing the packets on each of their queues.

At any time during the simulation, the user can pause the simulation by selecting the Pause item from the simulation menu. The simulation will then stop, allowing the user to inspect different statistics at each of the CommNodes, CommHosts and CommLinks using either the ViewResults item from the main menu or the PrintResults item from the main menu.

At any time during the simulation whether paused or running, the user can look at the Generated Traffic, the Terminated Traffic and the Network Delay matrices by selecting the Draw option from the title bar menu in each of those window matrices. To do this, the user positions the cursor over the title bar of the matrix, presses the left mouse button and selects the Draw option from the pop-up menu. Note that the Network Delay matrix shows accumulated network delay and not average network delay.

Illustrations of the display during a simulation and the possible print results and view results option are shown in Figures 3.7 through 3.11.

3.4.4 User Interaction Results

We have found that for the prototype network simulator that the user interaction techniques are very effective. A user can define a point-to-point packet switched network of medium complexity in a matter of minutes using these techniques. The user has very quick access to defining host-to-host traffic. While somewhat difficult in the prototype simulator, the user can also define node, host and link characteristics. The ability to see the network operate in terms of queue length at each node, host and link is also important to the intuitive understanding and visualization of the network operation. We feel that this ability to watch the network in operation can be a significant aid to understanding the behavior of complex networks.

3.5 Example of Simulation Results - Study of Routing Algorithms

This section describes an example of a simulation using the prototype network design tool. Figure 3.17 illustrates a simple point-to-point network. The example network consists of three hosts (H1 - H3), three switches (S1 - S3) and interconnection links. The hosts and switches are capable of handling 100 messages per second and the links are capable of handling 96 messages per second. Messages are fixed length. Simulations were run using both a flooding strategy and fixed route with alternatives strategy and with and without a single disturbance. Results are shown in Figures 3.18 through 3.22.

Figure 3.18 illustrates a plot of Queue Length versus Time for node S3 under a flooding strategy and with a disturbance. The host-to-host traffic was symmetrical and 3 messages per second, a relatively low value. At 2.005 seconds into the simulation, the link from S1 to S2 goes down and at 3.005, the same link comes up again. The plot in Figure 3.18 represents a secondary route between H1 and H2. No noticeable change is evident in the plot. This

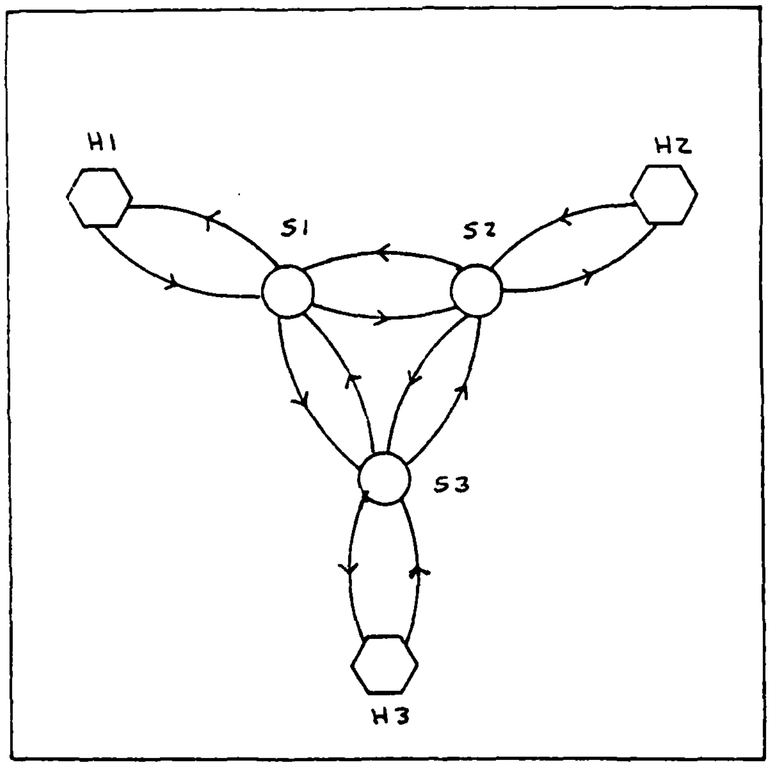


Figure 3.17 Topology of the Network Simulated

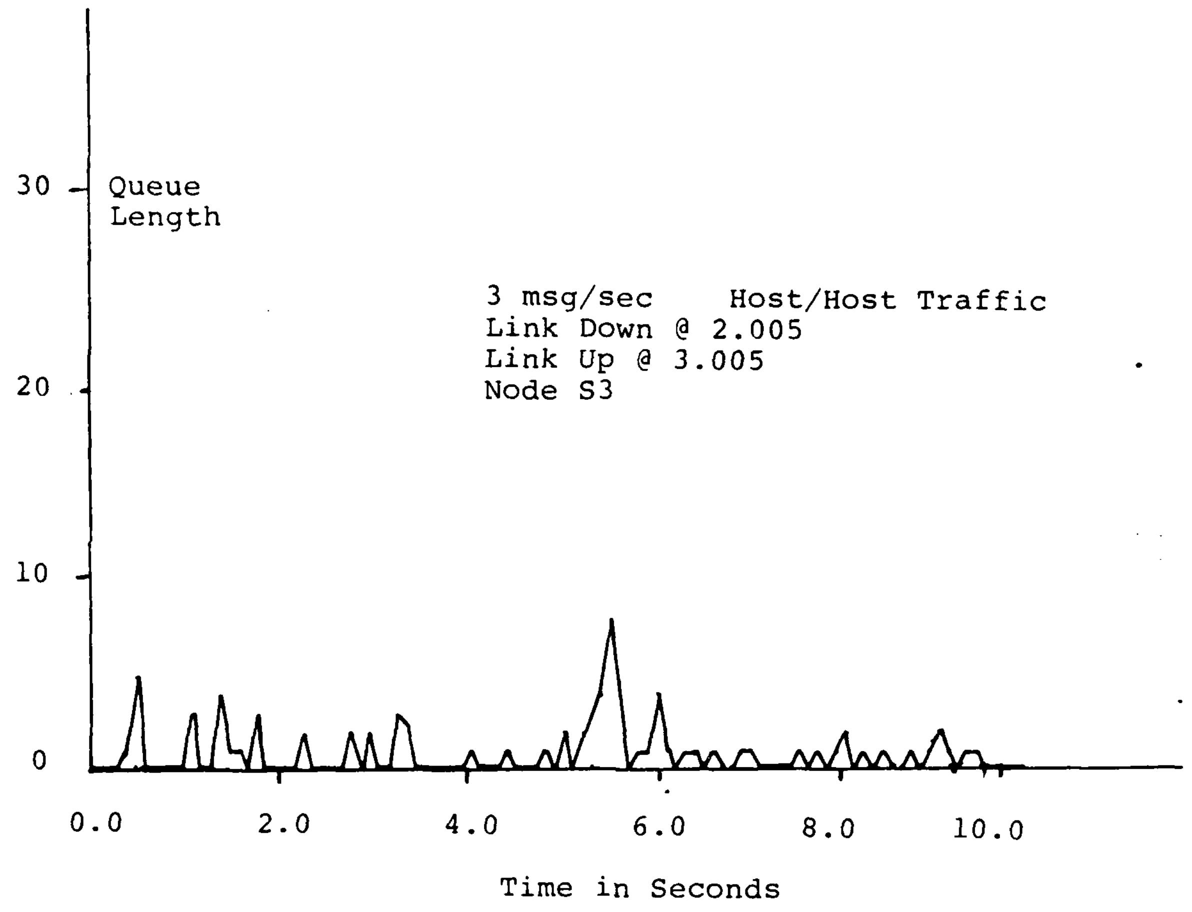


Figure 3.18 Queue length versus time at Node S3 with flooding algorithm

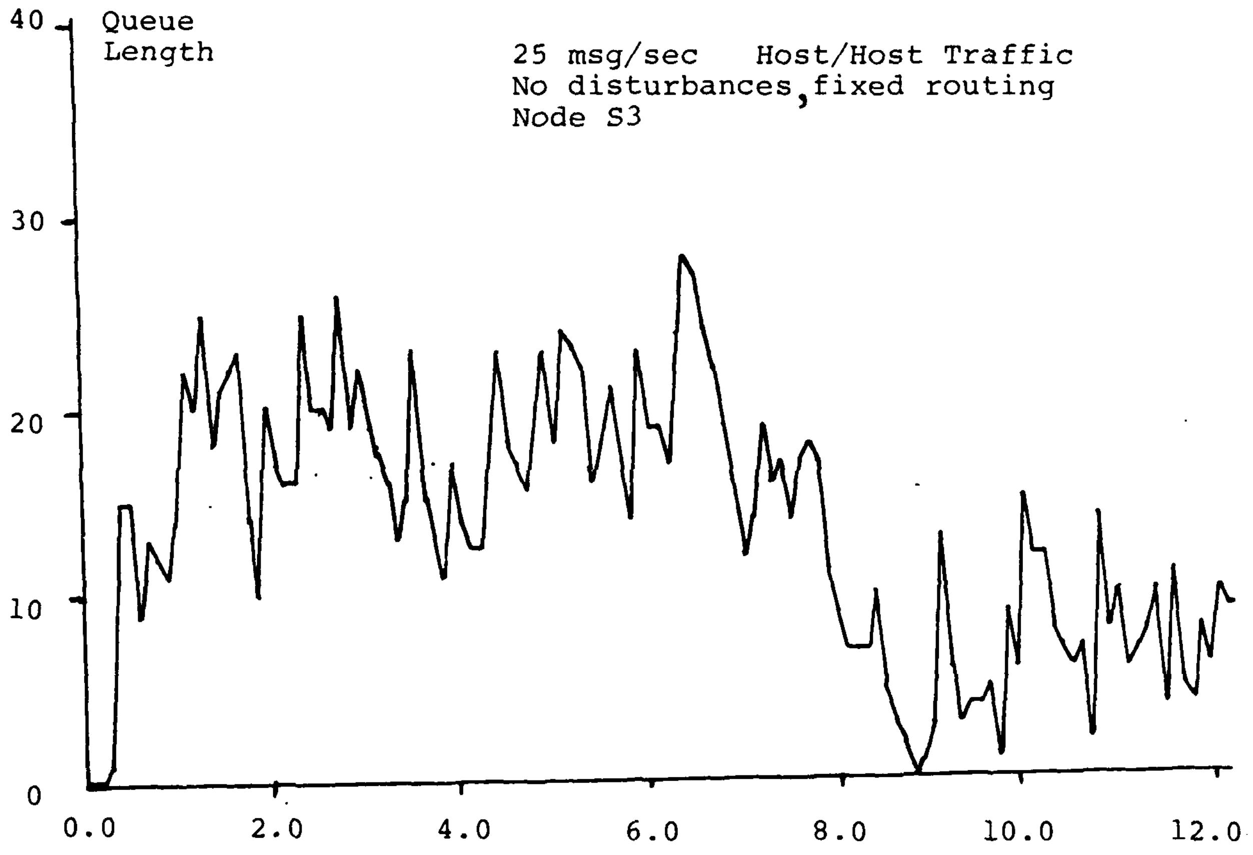


Figure 3.19 Queue length versus time at Node S3 with fixed routing

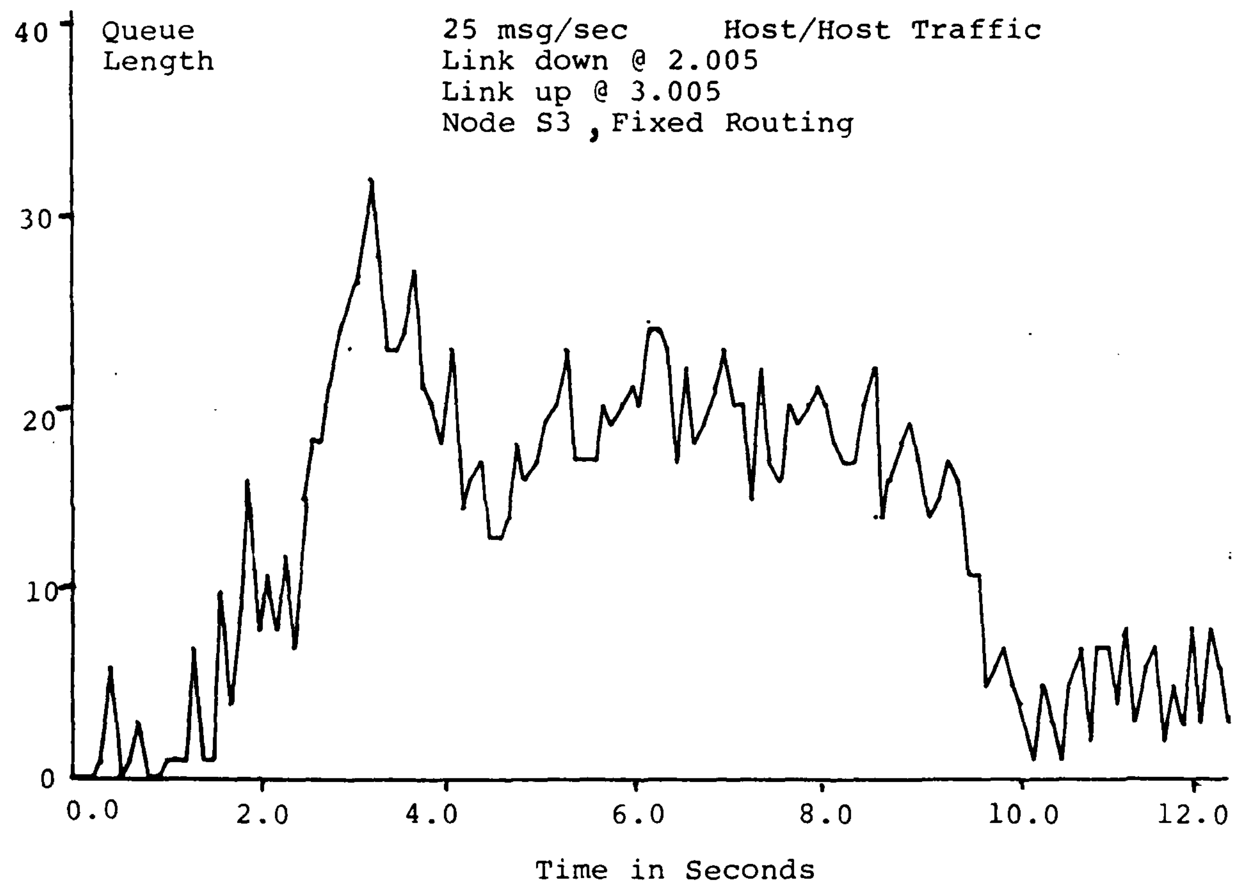


Figure 3.20 Queue length versus time at Node S3 with fixed routing

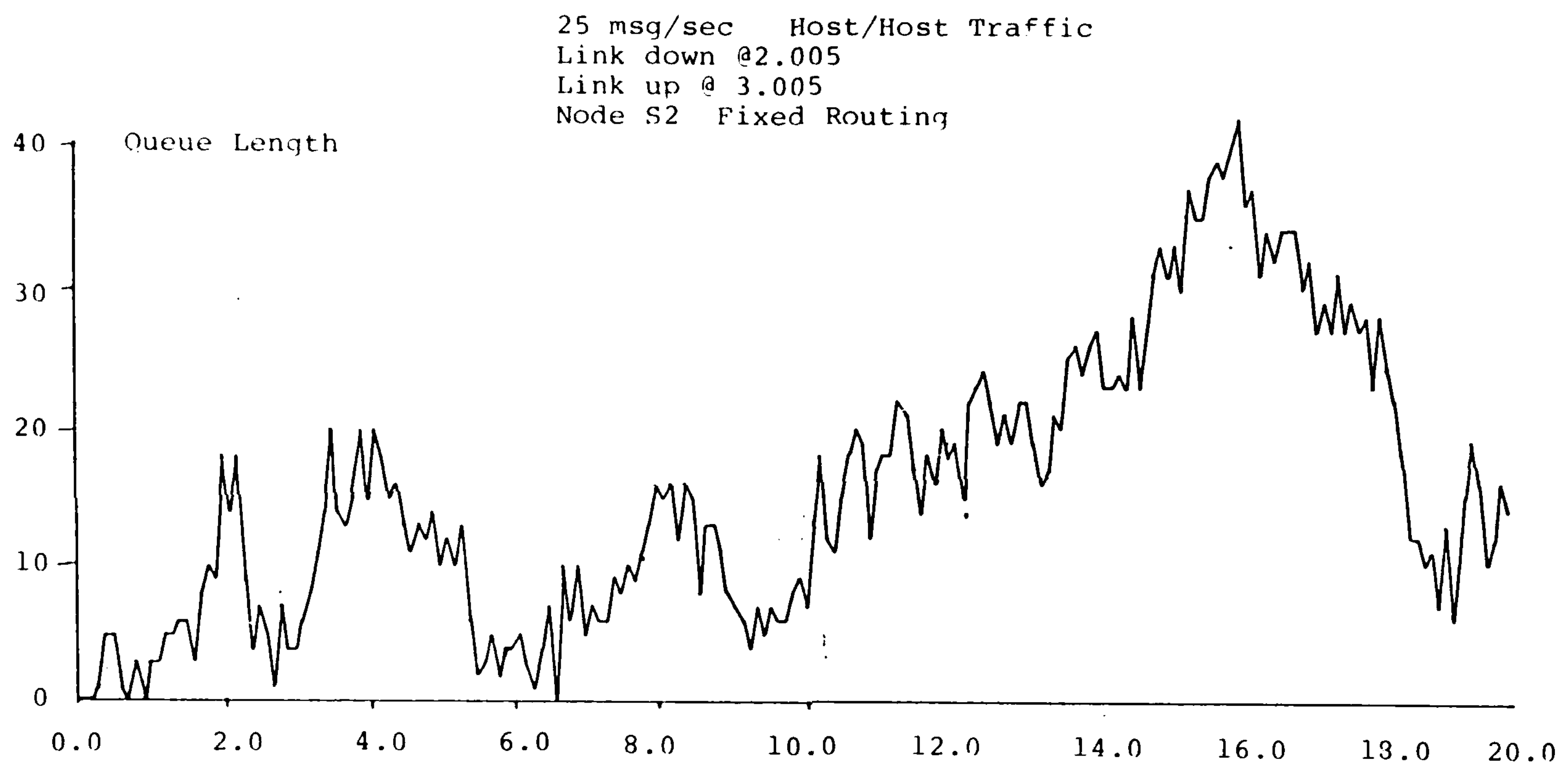


Figure 3.21 Queue length versus time at Node S2 with fixed routing

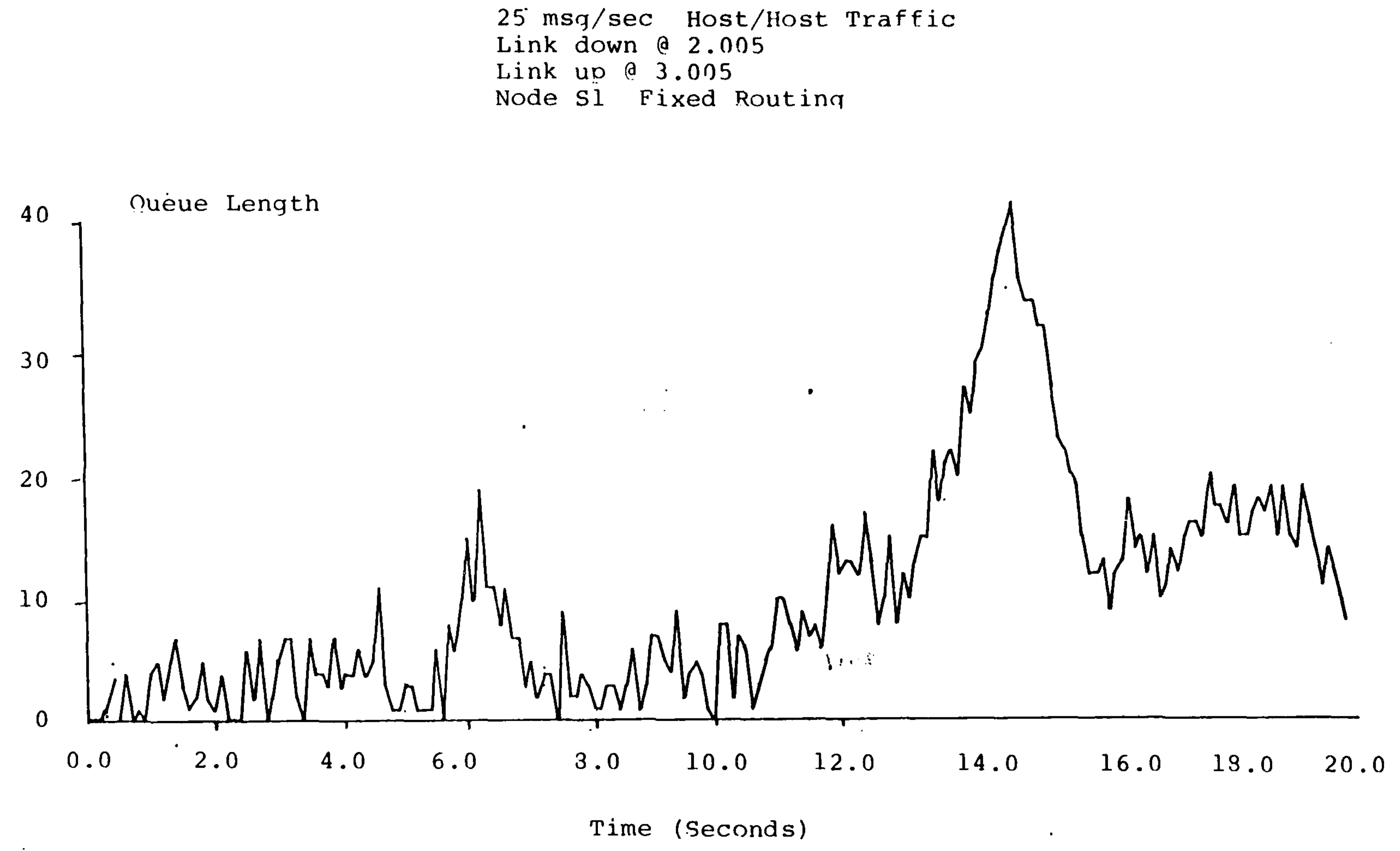


Figure 3.22 Queue length versus time at Node S1 with fixed routing

is expected as messages from S1 are always transmitted from S1 to S2 and S3. When the link from S1 to S2 goes down, messages are simply not sent from S1 to S2.

The second experiment was run with a fixed routing with alternatives strategy. The same network and disturbance was used. The host-to-host traffic was set to 25 messages per second. This repesents a heavily loaded network. Figure 3.19 illustrates a Queue Length versus Time plot for node S3. This represents a base simulation to compare to the disturbances described below.

Figures 3.20 through 3.22 show the reaction of the network to a disturbance. The link from S1 to S2 goes down at 2.005 seconds into the simulation and comes back up at 3.005 seconds. Traffic normally carried on this link is re-routed through S3, that is S1 to S3 to S2. In Figure 3.20, there is an initial peak in the Queue Length shortly after two seconds. While the disturbance only lasts for one second, the effect on S3 lasts for almost eight seconds.

Figure 3.21 illustrates the Queue Length at node S2. There is an initial drop in the Queue Length at two seconds into the simulation. This represents the absence of traffic from S1. Traffic returns to normal shortly after three seconds into the simulation. As messages are processed at S3, more messages (normally traffic plus re-routed traffic) are routed from S3 to S2. Therefore, the queue, at S2 begins to increase. Node S2 will be processing normal traffic plus the re-routed traffic. Thus, node S2 begins to overload. This overload peaks approximately 16 seconds into the simsulation and 14 seconds after the disturbance. Figure 3.22 shows the Queue Length at node S1.

Comparison of the simple routing algorithms studied support the following conclusions: The flooding algorithm is more robust than fixed routing strategies when disturbances occur in the network. Fixed routing strategies take much longer to recover from events such as link or node failures. On the other hand, the throughput of networks using flooding strategies will be considerably smaller than the throughput with fixed routing strategies.

This simple study was not intended to serve as a final evaluation of routing algorithms. Rather, it was presented to show the flexibility of the simulation approach and to illustrate how the dynamic behavior of a network can be observed using simulations. Detailed comparison of routing algorithms for survivable networks will be done in Phase II of this project.

3.6 Recommendations for Future Work

There are seven areas which require enhancement in a production version of the network design and simulation tools. Several of these features are well understood and simply require enhancements to the work that has already been performed. The remaining features are based on current research into network management and formal protocol definition which will require additional research and experimentation to arrive at a useful tool. In this section, we describe seven features that will be included in a production network simulator. These are:

1. User Interaction
2. Network Management
3. Internetwork Communication
4. Packet Radio
5. Integration of Analytic and Discrete Simulation
6. Additional Communication Link Types
7. Protocol Definitions and Implementations

3.6.1 Future User Interaction

The user interaction techniques developed so far allow rapid definition of a simple point-to-point communication network. The interaction techniques can be improved by providing additional means to define the network topology communication entity and to change parameters. The system should have available a number of common network topologies that the user can choose from as a base topology for a design. In addition, the network design tool should provide the user with some information as to the reliability and robustness of the network topology. Network topologies and criteria for evaluating networks are reasonably well known so that this is a simple implementation problem.

A second area to improve in the user interaction is the definition of communication parameters such as link capacity, bit error rate, node capacity, routing information and consistency checking, etc. This area also requires additional implementation work and should not prove difficult.

A third area for improvement is the presentation of results of the simulation. This can take the form of better print-outs, displaying results of the simulation in time plots, gathering additional statistics during the simulation, and real time presentation of the network behavior as disturbances occur and conditions change within the network. There should be no difficulty in providing these enhancements.

3.6.2 Network Management

One of the most important areas to include in a state-of-the-art simulator will be the area of network management. The area of network mangement includes new routing algorithms incorporating expert knowledge, the ability to reconfigure a network that has been damaged in some way and the ability to automatically recognize damaged parts of the network and issue repair commands for reconfiguration. While a network design tool should not provide the algorithms necessary for the network management function, the design tool must be able to provide for the evaluation of new network management techniques. Therefore, it will be necessary to include a capability within the network simulator for testing approaches to these network management techniques. Since network management at a high level (i.e., reconfiguring a network, rerouting large amounts of traffic, and recognizing faults in the network) requires a large amount of expertise, we expect that expert systems technology will have to be used to accomplish these tasks. Therefore, the network design tool must include a means for the network designer to define the expertise for management and provide for testing these new network management techniques.

There are two different problems in network management, each of which requires a different approach to reaching conclusions about the network. The first type is a top-level or goal-directed requirement such as connecting point A to point B through a network or number of networks. The goal in this case is to connect point A to point B and the job of the network manager is to find a series of connections to achieve that goal. The second type of reasoning that is required is a bottom-up approach primarily used in maintenance functions. As network status reports arrive at a network management center, the network manager must digest these reports and reach conclusions about the status of the network. The inference here is from fine, minute details about the network to larger conclusions such as a link being down or a node unavailable.

The area of expert systems is sufficiently well developed that each of these reasoning approaches can be included in a network design tool. The major problem in using this technology will be training the network designers to take advantage of it. To make this task as easy as possible, we propose to provide a number of typical management functions that can be tailored to a desired network.

3.6.3 Internetwork Communication

The third area to include in a network simulator involves internetwork routing. The simulator to date handles routing within a single network and does not provide the means for routing within multiple connective networks. The internetwork capability is tied very closely to the protocol definition feature of the simulator. It is the protocol which defines the ability to route messages from one network to another.

3.6.4 Packet Radio

The fourth area of enhancement is in the area of packet radio. Packet radio nodes are dynamic communication nodes. They have additional parameters which include position, velocity, transmission power and reception sensitivity. These parameters in turn, mean that the connections between packet radios become extremely dynamic, that the quality of the link reflected in bit error rate is dynamic, and that the routing tables for messages through the network are also dynamic. The inclusion of packet radio in a network simulation implies that the network designer must be able to specify the four parameters listed above, program those parameters for the duration of the simulation, and provide a model for the link quality achieved from the parameters. The protocols implemented in the network must also take into account the dynamic nature of the connections and routing within the network.

Building a simulator that allows the user to define packet radio systems should not be difficult. The major work is left to the network designer to specify the protocols for recognizing and adapting to the dynamic nature of packet radio systems. The network design tool must provide the tools necessary for the analysis of these systems.

3.6.5. Integration of Analytic and Discrete Simulation

In the present report, we have outlined two approaches to simulation. One, a straight forward event simulator with a convenient user interaction interface, and the second, an analytic approach to reducing the amount of simulation time needed to achieve acceptable results. A significant area of work is the integration of the analytic results presented in Chapter 4 of this report and the discrete simulation presented in the current Chapter. The needs of the analytic approach are fairly well understood, and this integration is an implementation task.

3.6.6 Additional Link Types

The production network simulator will include several different types of links beyond the point-to-point. We have already mentioned the inclusion of packet radio as one type of link to be added to the network simulator. We also plan to handle time division multiplexed links, communication buses, such as Ethernet, and satellite links.

3.6.7 Protocol Definition

Perhaps the most difficult area today in simulating a communications network is the definition and simulation of the protocol used in the network. To date, there is no convenient way to specify a communication protocol and simulate that protocol without first implementing the protocol for the simulation. This, of course, is a large amount of work to perform. The production version of the network simulator will include one or more ways for the network designer to define a network protocol in an interactive manner that does not require the designer to implement a protocol as a program. The approaches we have considered to date include the modeling of a protocol with finite state machines (automata), Petri networks, production rule systems or logic based systems, primitive protocol models and conventional programming procedures. Each of these approaches has several advantages, however, no single approach stands out above the others. The ease in defining a proposed network protocol is perhaps the most important item in a state-of-the-art network simulator.

3.6.8 Conclusion

We have outlined seven areas for enhancing the prototype network simulator. Five of these areas: user interaction, internetwork routing, packet radio, integration of analytic and discrete simulation and new communication link types are well understood and simply require the implementation to be performed. The areas of network management and protocol definition will require some research and experimentation to reach a point that it is convenient for a network designer to use.

4.0 COMPUTATIONALLY EFFICIENT SIMULATION APPROACHES FOR COMMUNICATIONS NETWORKS

Computationally efficient techniques for the simulation of data communication networks are investigated. Analytical techniques are reviewed and their limitations cited. Statistical problems in obtaining network performance measures are introduced. Problems with overcoming initial bias and sample correlation are shown to have a significant impact on the simulation efficiency. Techniques used to overcome these problems are discussed. Regenerative methods for simulation are introduced. Hybrid (combined analytic/simulation), variance reduction and perturbation techniques are discussed as methods to improve the computational efficiency of network simulations. A new variance reduction technique is introduced based on partitioning regenerative busy cycles into one group which contain only high frequency events and another group which contains low frequency (rare) events. For the case studied here, a reduction in computational effort of a factor of approximately 5.5 was obtained. Recommendations for future research for developing computationally efficient techniques for network simulation are presented.

4.1 Introduction

Computer communications systems are becoming larger and more complex. Predicting the system performance based solely on analytic models is becoming increasingly difficult, if not impossible. For large complex computer communications networks, simulation is playing an increasingly important role in performance prediction. Even though network simulation can be used to model a system in minute detail, the computational aspects of the simulation method currently limit its applicability. Simulation has been primarily used for validation of approximate analytic approaches or for performance prediction in specific case studies [12]. The key challenge in the development of performance prediction tools for large computer communication networks is to be able to devise computationally efficient simulation techniques.

The purpose of this section is to examine existing techniques for improving the computational efficiency of network simulation. These techniques are discussed relative to their applicability to the Packet Communications Network Synthesis and Analysis System (PCNSAS) being developed by ST*AR Corporation under the Defense Small Business Innovation Research Program (SBIR). The PCNSAS is to be a computer-aided design tool for general packet communication networks (PCN). PCNSAS will consider all aspects of PCN, e.g., link capacity assignment, routing, flow control, protocols, link quality and link availability. The intended generality of PCNSAS has a significant impact on the classes of computationally efficient techniques that can be applied.

The primary results of this investigation is a design plan for the development and implementation of efficiency improving techniques for PCNSAS.

To provide an adequate background, an overview of existing analytic techniques will be provided in Section 4.2. The emphasis of this section will be on enumerating the situations where analytic techniques are and are not applicable. Section 4.2 will demonstrate that analytic techniques are not suitable for many cases of interest. Therefore, we must resort to simulation for obtaining performance predictions. In Section 4.3, we will discuss the statistical nature of network simulation. The statistical problems that are encountered in network simulation will be presented. It is in this section that we will present the scope of the statistical difficulties that will be encountered in obtaining reliable performance predictions using PCNSAS.

Common aproaches for calculating confidence intervals on the network performance measurements will be discussed in Section 4.4. The width of these confidence intervals provide an indication of the reliability of the simulated performance predictions. Improving the computational efficiency of a network simulation can be viewed as reducing the width of the confidence interval for the same computational cost or, stated another way, reducing the computational cost while maintaining the width of the confidence interval.

The computational efficiency of network simulation is improved through the skillful use of prior knowledge of how the network operates. There are two primary approaches to integrating prior knowledge into a network simulation. The first technique uses analytic methods to model some specific theoretically tractable aspects of the network while employing simulation to model the remaining segments of the system. The analytic and simulation submodels are combined to form what is called a hybrid simulation model. (See [13] for a detailed definition and classification of hybrid simulations).

The second approach to integrating prior knowledge of the network operation into the simulation model is to modify the simulated input random processes in some known way which produces the desired performance measures with a reduction in variance. These methods are called Variance Reduction Techniques (VRT). (See [14] for a detailed discussion of these methods). Both hybrid and VRT will be discussed in Section 4.5. This discussion will be in the form of reviews of previous applications of these techniques. Through these reviews the strengths and weaknesses of each approach will be identified. The application of each approach to the PCNSAS will be considered.

The review of existing approaches was used as the basis for the fomulation of a plan to incorporate computationally efficient techniques into PCNSAS. Because of the generality requirement of PCNSAS, the efficiency enhancing techniques discussed in Section 4.6 will focus on several approaches. One approach will be applicable to packet radio networks. Here a technique to model a large packet radio network as a combination of a small number of primary stations (to be simulated in detail) and a large number of background stations (to be modeled as one source alternating between a busy and idle state) will be outlined. The second approach will be based on the concept of "rare" events, e.g., the sudden degradation of link quality. An approach will be outlined where the occurrence of rare events is controlled to reduce the computational requirements of the simulation. Also considered for further study are general hybrid, perturbation analysis, and control variate techniques. We feel that these approaches offer the greatest promise for improving the computational efficiency of PCNSAS while maintaining its flexibility.

4.2 An Overview of Analytic Techniques for Communications Network Analysis

The purpose of this section is to present the common analytical results which are available for the analysis of networks. The results will not be rederived here, however, their applicability and underlying assumptions will be clearly noted.

The performance criteria that are of interest in network analysis will be discussed first. The common assumptions used for the traffic in data communications networks will be presented next. This will be followed by the known results for a single server queueing system. Networks of queues are widely used to model computer communications systems. Analytic results for networks of queues will be discussed. The methods described in this section are well documented [1, 4, 15, 16, 17, 18, 19].

4.2.1 Computer Communication Network Performance Criteria

The performance of a computer communication network depends on your perspective. As discussed in [20], the network user is primarily interested in how quickly his message is delivered to its destination. That is, he wants the time delay per message to be small. A critical performance criteria is then message delay. The manager or owner of the network is interested in how efficiently his network resources are utilized. Thus, throughput (the time the network is being utilized sending user information divided by the total time) is another critical parameter. Delay and throughput are the two most common performance criteria of communications networks. The network designer is concerned with how much memory (the size of the message buffer) is required in various nodes in the network and how fast the network node processors must be to meet the design objectives. Thus, the designer is mainly concerned with buffer and protocol efficiency.

Throughout the remainder of this report, the three primary performance criteria will be: 1) delay, 2) throughput, and 3) buffer efficiency. Note that these quantities are not independent.

4.2.2 Common Traffic Assumptions in Data Communications Networks

Traffic in data communications networks is generated by a variety of sources, e.g., terminals, file servers, print servers, analog sources, i.e., digitized voice or video. Two values typically characterize data traffic: 1) the message length (in bits, bytes or normalized to transmission speed of the communications link in sec), 2) the time between message arrivals. Both the message length and interarrival time are random variables. Most theoretical analysis are restricted to cases where the interarrival time probability density function (p.d.f.) is exponential. When the interarrival time is exponential, the arrival process is Markovian and the number of messages arriving in any fixed finite time interval follows a Poisson probability mass function (p.m.f.). Analytic models usually do not restrict the message length p.d.f. However, the general assumption is that the message length p.d.f. is also exponential.

Unlike telepone traffic which has been extensively studied and accurate traffic models constructed, data networks are relatively new and rapidly evolving, thus accurate models for network data traffic do not exist. (Some measurement programs have recently been performed [21, 22]). Lacking accurate traffic models, most analytic techniques rely on the assumption of exponential interarrivals of messages and to a lesser extent exponential message lengths.

4.2.3 Overview of Results for Single Server Queueing Systems

The purpose of this section is to present some common results for a single server queue. Consider messages arriving at the queue in a random fashion with random lengths. Let

$X_n \overset{\Delta}{=}$ service time of nth arriving message (in sec) or length of the nth arriving message (in sec) for a given transmission link capacity in bps.

$C_n \overset{\Delta}{=}$ the nth message arriving to the system

and

$T_n \overset{\Delta}{=}$ interarrival time between C_{n-1} and C_n message

As discussed in [15], X_n represents the amount of work (resources) the nth message requires while T_{n+1} is the time between the arrival of this and the next message. For all tractible analysis X_n and T_n must be statistically independent. Let

$$U_n = X_n - T_{n+1}$$

then for the system to remain stable, i.e., the number of messages waiting for service remains finite, U_n must be negative on the average. Thus, on average there will be more "breathing space" between messages than demand for system resources. This condition can be stated in terms of the traffic intensity, ρ offered to the buffer for stationary systems. Let

$$\rho = \lambda L$$

where $\lambda \triangleq$ average message arrival rate

$$= E\{1/T_n\}$$

and

$$L \triangleq E\{X_n\}.$$

The buffer length will remain finite if

$$\rho < 1$$

One of the primary goals is to predict the properties of the message waiting time. Let

W_n = waiting time for nth customer or message

Then we can recursively find (simulate) a sequence of waiting times as

$$W_{n+1} = (W_n + U_n)^+$$

where

$$(X)^+ = \max(0, X)$$

Note $W_n + U_n$ will be negative if the nth message finishes before the $n + 1$ message arrives and thus, the $n + 1$ message will experience no waiting time. Note W_n does not include service time. Unfortunately, an exact general solution for even the average waiting time is unknown [15]. However, if the arrival process is Markovian, then an exact solution is known. Specifically, the expected value and variance on the waiting time* is given by

* This analysis assumes no blocking and thus, unlimited buffer capacity.

$$E\{W\} = \frac{\lambda\, E\{X^2\}}{2(1 - \rho)}$$

and

$$Var\{W\} = (E\{W\})^2 + \frac{\lambda\, E\{X^3\}}{3(1 - \rho)}$$

Thus, the average waiting time is function of the second moment of the message length distribution while the variance is related to the third moment.

As previously discussed the most common assumption for the messsage length distribution is exponential. Thus for exponential message lengths where

$$E\{X^2\} = 2(E\{X\})^2$$

the average waiting time becomes

$$E\{W\} = \frac{\rho\, E\{X\}}{1 - \rho}$$

Noting that

$$E\{X^3\} = 6(E\{X\})^2$$

then for exponential message lengths the variance on the waiting time becomes

$$Var\{W\} = \frac{\rho}{1 - \rho} (E\{X\})^2 \left[\frac{\rho}{1 - \rho} + 2 \right]$$

The average delay or time a message is in the system is the sum of the average message length (in sec) and the average waiting time $E\{W\}$, i.e.,

$$E\{T\} = E\{X\} + E\{W\}$$

where

$E\{T\}$ = average delay

The quantity $E\{W\}$ represents the cost of sharing the buffer with other messages.

Buffer occupancy is another primary performance criteria in network design and analysis. There is a broadly applicable relationship between the average delay and average buffer occupancy known as Little's formula. Little's formula states that

$$E\{N\} = \lambda E\{T\}$$

where

N = number of messages in buffer

Little's formula holds for non-Poisson arrivals and general message length distributions as well as arbitrary order of message service. From [4], the assumptions required for Little's formula to be valid are:

1. At some time the buffer becomes empty with probability 1.

2. There is a certain stationarity in that an arrival to an empty system begins the queueing process afresh.

3. While the buffer is occupied, the mean delay and mean time between arrivals is finite.

4. The expected number of messages in the buffer between the time the buffer is empty until it becomes empty again (this interval is called a busy cycle) is finite.

Little's formula is very powerful in that given the average number of messages in the system and the average arrival rate, the average delay can be predicted for a very wide range of cases. (Note: using the standard assumptions on the arrivals and message lengths an exact probabilistic model for the number of messages in the buffer can be found).

The above results summarize the common characterization of a single server queues. Much more detailed models exist which consider the service discipline and priority structure for a single server queue (e.g., see [4, 15 and 16]). Computer communications networks are comprised of networks of queues, in the next section analytic models of networks of queues will be considered.

4.2.4 Techniques and Assumptions for the Analysis of Network of Queues

Data communication networks can be modeled as a set of interconnected queues. In this section, we will introduce the techniques for analyzing the performance of networks of queues. It should be noted before proceeding that as discussed in [18] "the mathematical intractability of general queueing networks requires great simplification to obtain analytic solutions and/or make simulations economically feasible." The purpose of this section will thus be to explain the analytic technique and emphasize its applicability, i.e., its underlying assumptions.

As discussed in [4], a node in a store-and-forward communications network accepts messages from external and internal sources, a nodal processor examines each message for errors and to determine its route, i.e., the correct outgoing link for that message. Messages are buffered before being transmitted on each link. Thus, there is one buffer, queue, for each outgoing link. Typically, the nodal processor is much faster than the transmission rates and thus the buffering and processing times are usually ignored.* The result is a network model shown in Figure 4.2.1.

Clearly in such a network the service time (message length in sec) is not independent between node 1 and 2. The service time in a packet communication is proportional to the message length in bits. As the message travels through the network its length in bits is constant and thus the service time at various nodes in the network are dependent. Further, the interarrival time of two successing messages over a given link will be dependent upon their service time at the previous node. Thus, the interarrival process is not independent

* Note this assumption is not true for some local area networks.

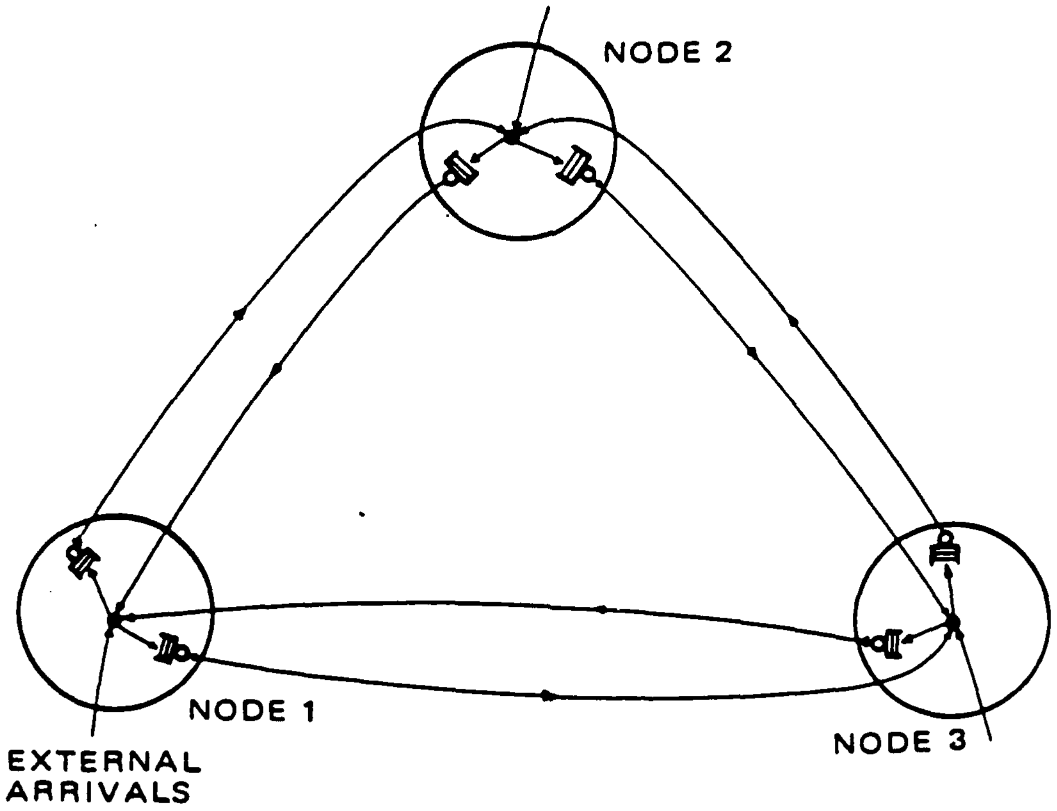


Figure 4.2.1 Three-node store-and-forward message-switched network (from [5])

of the service time process. These properties are incompatible with the necessary assumptions for analytic tractibility [18].

The key assumption for analytic tractibility is independence and was made by Kleinrock [15]. Under the independence assumption service times are chosen independently at each node. Clearly the assumption requires justification. As discussed in [18] there are three randomizing mechanisms justifying independence:

1. Error recovery decorrelates "virtual service times" and message lengths.

2. The splitting and mixing of various independent input streams into a single outbound link.

3. Average buffer and delay statistics are not sensitive to the service dependencies.

The message independence assumption was tested using simulation and it was found that as few as two to three incoming or outgoing links resulted in sufficiently independent service times. Having made the independence assumption and further assuming all external arrivals are Poisson and have exponential message lengths, the queueing network can be decomposed into N individual, independent M/M/1* problems where N is the number of communications links in the network. The analysis of such networks is straight forward (see [23]). The average network message delay is found to be

$$E\{T\} = \sum_{i=1}^{N} \frac{\lambda_i}{\gamma} \frac{E\{X_i\}}{(1 - \rho_i)}$$

where

γ = total external message arrival rate

* M/M/1 is queueing theory notation indicating Markov arrivals, and exponential message length and one server.

λ_i = arrival rate for link i

ρ_i = traffic intensity for link i

$E\{X_i\}$ = average message length in second on link i.

As discussed in [15], the above result and assumptions can be used to perform optimum designs in four areas: 1) link capacity assignment, 2) flow assignment, 3) capacity and flow assignment, 4) topology, capacity and flow assignment.

The common assumptions for network analysis are summarized in Table 4.2.1. The state-of-the-art in analytic modeling can thus be stated as [18].

1. Exact analysis of a single protocol in a point-to-point or multipoint environment is feasible to a great degree of detail.

2. Networks are hard to analyze and the modeler is forced to reduce the problem to exponential systems. This always requires strong assumptions such as message independence.

Table 4.2.1

1.	Message independence.
2.	Poisson arrivals.
3.	Identically distributed message lengths.
4.	No channel errors and message retransmissions.
5.	Unlimited buffer capacity.
6.	No adaptive routing.

Common Assumptions for Network Analysis

3. Even where message independence is made, meshed topologies may turn out to be too difficult for analysis. In a protocol analysis, one may simplify the topology without significant loss of insight.

The above summary clearly indicates that simulation will be the only performance analysis tool suitable in many cases of interest. The remainder of this report is devoted to investigating the statistical aspects of network simulation.

4.3 Statistical Problems in the Simulation of Data Communications Networks

The purpose of this section is to introduce the concept of simulating data communications networks for performance analysis. The design of simulation experiments and the interpretation of their results is a nontrivial problem. The statistical problems encountered in data communication network simulation will be reviewed here. There are many texts devoted to discrete event simulation e.g., [14, 24, 25,] in which a more complete discussion can be found.

4.3.1 Stochastic Discrete Event Simulation for Data Computer Network Analysis

Stochastic discrete event simulation must be explained by discussing the stochastic and discrete event aspects of the process.

Stochastic simulation involves generating sequences of pseudorandom numbers with specified distributions to represent the system input, for example, message lengths and interarrival times. These sequences of numbers are used to drive a model of the queueing network, the model typically is an algorithmic description of the network operation implemented on a computer.

The term discrete event refers to the nature of the algorithmic description of the queueing model. Queueing systems are dynamic, the length of queues changes with time. However, the events which change the length of the queue for example can only occur at a countable number of discrete points in time. That is, we are not concerned with the time in between events, only the actions taken upon their occurrence. In discrete event simulation, the dynamic behavior of the network is obtained by using pseudorandom sequences of numbers to determine how to change the state of the network in accordance with a well defined set of rules which model the network operation at discrete points in time.

4.3.2 Statistical Analysis of Simulation Output

The analysis of the results of a discrete event simulation is not easy. The following description of the statistical nature of discrete event simulation was obtained from [19] modified to emphasize data communications network instead of computer system performance. Discrete event simulation is controlled statistical experimentation. Models are driven by random input sequences, such as message lengths and produce random output sequences, such as delays from which estimates of the delay distribution and/or its moments are obtained. If the simulation is rerun under identical conditions (but driven by different statistically independent input sequences), the output estimates will be different, often dramatically so. Thus, as in the case with measurements, sound statistical methods are required to interpret the results of simulation models.

Because simulation outputs are random, it is important to assess the amount of variability in estimates that is due purely to random sampling effects. In addition to assessing statistical accuracy, it is important to be able to adjust the length(s) of the simulation run(s) so as to obtain estimates of specified accuracy. These two problems of accuracy assessment and run length control can be addressed through the use of confidence intervals. Suppose μ is some unkonwn output characteristic of the model to be estimated, e.g., the mean steady state delay time. An interval (L, U) with random endpoints L and U is said to be a $100(1 - \alpha)$ percent confidence interval for μ if Prob $\{L < \mu < U\} = 1 - \alpha$. For $(1 - \alpha)$ close to one, there is a high probability that the unknown parameter μ is between L and U. The confidence interval is usually formed in such a way that a point estimate $\hat{\mu}$ of μ lies in the interval and is frequently the midpoint of the interval. Formation of such a confidence interval generally requires an estimate of the variance of $\hat{\mu}$. The width of the interval U-L is then a measure of the accuracy of $\hat{\mu}$. The narrower the interval, the more confidence can be placed in the estimate. Simulation run length control algorithms can be developed that allow a model to be run until confidence intervals of suitable accuracy (as defined by the analyst) are obtained.

Classical statistical techniques can be applied when estimating transient characteristics since multiple independent replications can be performed thereby generating i.i.d (independent and identically distributed) observations. This is because transient effects are typically of short duration. However, simulation studies of queueing network models for network performance often involve estimating steady state characteristics. There are a variety of procedures for generating confidence intervals for steady state characteristics. Standard statistical approaches based on i.i.d observations cannot be directly applied since simulation output sequences are usually both correlated and nonstationary. The correlation arises because of queueing; if the waiting time of a message at a node is large, then it is likely that the waiting time of the next message will also be large. Nonstationarity is a consequence of the model's initial conditions. If the simulation is started with all nodes transmitting a message simultaneously, then artificial queueing delays will result. It should be emphasized here that there are three basic statistical problems that exist in predicting the performance of network using simulation:

1. Starting rules, i.e., what is an appropriate initial state for network or how can the effects of the initial transient be removed.

2. Stopping rules, i.e., how long to run the simulation.

3. Correlated samples, i.e., observations obtained from the simulation will be correlated.

To further quantify these problems, suppose we let $X_1, \cdots, X_N$ be the delay time output sequence of N messages generated by the simulation and that we are interested in estimating the mean steady state delay time $\mu = \lim_{n \to \infty} E(X_n)$. The usual estimate for μ is the sample mean

$$\hat{\mu} = (1/N) \sum_{n=1}^{N} X_n$$

An initial transient is introduced into a simulation by the selection of an initial state. During this transient, that is, for small n, $E(X_n) \neq \mu$ which in general means that $E(\hat{\mu}) \neq \mu$. The typical approach for dealing with this problem, which is also called the problem of the initial transient, is to determine an N_0 such that $E(X_n) \approx \mu$ for $n > N_0$, then delete the observations before N_0 and then estimate μ by

$$\hat{\mu} = (1/(N - N_0)) \sum_{n=N_0+1}^{N} X_n$$

This will be an approximately unbiased estimate of μ. Schruben [26] has proposed statistical tests for stationarity that can be used to test the adequacy of an N_0.

The other difficult problem is dealing with the autocorrelation. To explain the problem which arises from the correlation of the observed samples, we will first consider $X_1, \cdots, X_m$ to be m i.i.d. random variables with $E\{X\} = \mu_X$ and $\mathrm{Var}\{X\} = \sigma_X^2$. Our primary objective is to estimate μ_X. It is well known that the sample mean (for m samples) given by

$$\overline{X}(m) = \frac{1}{m} \sum_{i=1}^{m} X_i$$

is an unbiased estimate for μ_X. Further is is known that the variance for this estimator is

$$\sigma_{\overline{X}}^2 = \frac{\sigma_X^2}{m}$$

An unbiased estimator for $\sigma_{\overline{X}}^2$ is obtained by replacing $\sigma_{\overline{X}}^2$ by $S_X^2(m)$ in the above expression where

$$S_X^2(m) = \frac{1}{m-1} \sum_{i=1}^{m} (X_i - \overline{X}(m))^2$$

is the sample variance. That is, we estimate the variance of the sample mean as

$$\hat{\sigma}_{\overline{X}}^2 = \frac{S_X^2(m)}{m} .$$

A confidence interval for $\overline{X}$ would be formed by using the above estimate for its variance.

Unfortunately, the output data from a network simulation will be correlated. Consider the measurement of packet delay where if the ith packet experiences a large delay because of heavy traffic the i + 1 packet will with a high probability also experience a large delay. So not only are packet delay correlated, they are in general positively correlated.

Define a correlation coefficient between X_i and X_{i+j} as

$$\rho_{ij} = \frac{E\{(X_i - \mu_i)(X_{i+j} - \mu_{i+j})}{\sigma_i \, \sigma_{i+j}}$$

For a covariance stationary process

$$\mu_i = \mu_{i+j} = \mu_X$$

and

$$\sigma_i = \sigma_{i+j} = \sigma_X$$

so a correlation coefficient ρ_j can be defined independent of i. Again, we with to estimate μ_X and again the sample mean is an unbiased estimate for μ_X. But $S_X{}^2(m)$ is no longer an unbiased estimate for σ^2. In fact, it can be shown that [55]

$$E\{S_X{}^2(m)\} = \sigma_X{}^2 \; [\; 1 - \frac{2 \sum_{j=1}^{m-1} (1 - j/m) \, \rho_j}{m-1} \;]$$

In network simulation, $\rho_j > 0$, i.e., a positive correlation, and thus $S_X{}^2$ will have a negative bias and $E\{S_X{}^2\} < \sigma_X{}^2$. Thus if $S_X{}^2/m = \hat{\sigma}_X{}^2$ is used to form a confidence interval for $\bar{X}$ its width will be underestimated. Intuitively, it is reasonable to expect that if the samples are positively correlated, that is, they have a strong linear relationship, then the change from one sample to the next will be small and the sample variance will underestimate the true variance. The effect of the sample correlation on the simulation run length is illustrated in Table 4.3.1 (from Heidelberger). In this table, the required number of simulated packets N to obtain a 90% confidence about the mean is shown as a function of the traffic intensity. At low loads a reasonable

ρ	$E[W]$	σ^2	N
0.10	0.111	0.375	8,200
0.20	0.250	1.39	6,020
0.30	0.429	3.96	5,830
0.40	0.667	10.6	6,430
0.50	1.00	290	7,850
0.60	1.50	88.5	10,600
0.70	2.33	335	16,700
0.80	4.00	1,976	33,400
0.90	9.00	35,901	119,000
0.95	19.0	607,600	455,000
0.99	99.0	3.95×10^{8}	1.09×10^{7}

N = Number of customers that must be simulated for a 90% confidence interval for $E[W]$ to have a half length of $0.10E[W] = (1.645\sigma/0.10E[W])^{1/2}$; μ = service rate = 1; λ = arrival rate; $\rho = \lambda/\mu$; $E[W] = \lambda/\mu(\mu - \lambda)$.

Table 4.3.1 Sample sizes for the M/M/1 queue

number of packets are required. However, as the traffic intensity increases, the effect of the sample correlation also increases and thus the required number of simulated packets increases. Sample correlation cannot be ignored in generating confidence intervals since when positively correlated observations are assumed independent, severe underestimation of the variance and confidence intervals (much too narrow) would result. In the following section techniques for overcoming these problems will be discussed.

4.4 Common Solutions to the Statistical Problems in Data Communications Network Simulation

There are several common techniques used to overcome each of the difficulties (starting, stopping rules and correlated samples) encountered in network simulation. The purpose of this section is to discuss these techniques. Several of these techniques are based on the regenerative nature of the random processes involved in the queueing mechanism, these regenerative processes will also be reviewed.

4.4.1 Regenerative Processes in Simulation of Queueing Networks

Many queueing systems exhibit a regenerative structure. Suppose a message arrives at time t_0 and finds the system empty. This lucky message experiences no queueing delay. Let t_1 be the next time at which a message arrives with the system empty. For stable systems, i.e., $\rho < 1$, the difference $t_1 - t_0$ is finite [28] with probability 1. This difference is called the regeneration or busy cycle. The behavior of the system from t_1 on in time is a probabilistic replica of the system from t_0 on. The above is true for G/G/s queues, (i.e., queues with general arrival and message length p.d.f.'s and s servers). The importance of this regenerative nature is that comparable random variables defined over different busy cycles are statistically independent and identically distributed (i.i.d.). Regenerative techniques have been used in discrete event simulation [29, 30, 31].

4.4.2 Techniques for Dealing with the Initial Transient Problem in the Simulation of Data Communications Networks

Start up policies are needed to establish the initial conditions for the system. Clearly, if the regenerative method is used then all queues are initially empty and the problem goes away. The usual approach, however, is to start the simulation in some convenient state and to allow an arbitrary "warm up" period to pass before retaining output performance measures for analysis. This procedure is known as output truncation. Output truncation is the most common approach because it is simple and easily implemented. As previously discussed statistical tests for stationarity can be used to determine an adequate warm up period (see [32] for a survey of research on the start up problem in simulation). As discussed in [33] truncation techniques give no assurance that the bias caused by the initial transient will be effectively controlled. Further, simply throwing out data collected at the beginning of each run is wasteful especially if the warm up period is long. Because of its simplicity and ease of implementation, truncation techniques dominate. The topic of output truncation is treated in most texts dealing with discrete event simulation [24, 25, 5].

4.4.3 Techniques for Dealing with the Correlation of Output Sequences in the Simulation of Data Communications Networks

As discussed in [19], there are two general approaches to dealing with the correlation when generating confidence intervals.

1. Avoid it by organizing the simulation output into i.i.d. observations.

2. Estimate the correlation and compensate for it.

There are three methods to avoid the correlation. The first method is to use independent replications. This has the advantage of simplicity but the disadvantages of being sensitive to the effects of the initial transient and wasteful of data if the transient portion is discarded from the beginning of each replication. The second method is called batch means which operates on a single run. The sequence of packet delays is divided into long blocks, or batches, of length B so that the means of the batches are approximately i.i.d. If B is so chosen, then classical statistical methods can be applied. However, selection of an adequate B is a difficult statistical problem and there is currently no satisfactory method.

The third method is called the regenerative method which is based on the fact that some stochastic sequences, called regenerative processes, contain regeneration points which delimit the sequence into i.i.d. blocks of random length. We feel that the regenerative method can be effectively used in data communications network simulation. Because of its potential importance the regenerative method will now be discussed in detail.

Let X_n $(n > 1)$ be a regenerative process. Let α_j denote the number of messages processed in the jth busy cycle. Further, let $f(\cdot)$ be a measureable function such that the performance quantity of interest is $r = E\{f(X_n)\}$. Define a measurement made over the jth busy cycle as

$$Y_j = \sum_{\substack{i \text{ in} \\ j\text{th cycle}}} f(X_i)$$

For X_i representing the message delay Y_j is simply the sum of all delays of the packets processed in the j^{th} busy cycle. The fundamental properties of the regenerative processes are:

1. The sequence (y_j, α_j) consist of independent and identically distributed random vectors.

2. For finite X, $r \triangleq E\{f(X_n)\} = E\{Y\}/E\{\alpha\}$

The statistical estimation problem can now be stated as: Given a sequence of measurements from successive busy cycles (y_j, α_j), estimate the ratio $r \triangleq E\{\ell\}/E\{\alpha\}$ (note we are not trying to estimate $E\{Y/\alpha\}$). The classical estimator for r is $\hat{r}$ given by

$$\hat{r} = \frac{\frac{1}{n}\sum_{j=1}^{n} Y_j}{\frac{1}{n}\sum_{j=1}^{n} \alpha_j} = \frac{\overline{Y}}{\overline{\alpha}}$$

where n = number of observed busy cycles.

This estimator for r is strongly consistent and generally biased. A set of alternate estimators have been developed in an effort to reduce this bias (see [29] for details).

A $100(1-\delta)\%$ confidence interval for $\hat{r}$ (when n is large) can be obtained using the central limit theorem. Let $V_j = Y_j - r\alpha_j$. Clearly, the V_j's are statistically independent and identically distributed with zero mean. Denote the average of V_j, Y_j and α_j, by $\overline{V}$, $\overline{Y}$, and $\overline{\alpha}$ respectively where

$$\overline{Y} = \frac{1}{n} \sum Y_j$$

$$\overline{\alpha} = \frac{1}{n} \sum \alpha_j$$

$$\overline{V} = \frac{1}{n} \sum v_j$$

It follows that

$$\overline{V} = \overline{Y} - r\,\overline{\alpha}$$

Letting $\sigma_V^{\;2} = \mathrm{Var}\{V_j\}$ then

$$\sigma_{\overline{V}}^{\;2} = \sigma_V^{\;2}/n$$

and

$$Z = \frac{\overline{V}}{\sigma_V \sqrt{n}}$$

is approximately a standard normal random variable. Extending this argument using $\overline{V} = \overline{Y} - r\,\overline{\alpha}$ we relate z to the estimated ratio $\hat{r}$

$$Z = \frac{\overline{Y} - r\,\overline{\alpha}}{\sigma_V \sqrt{n}} = \frac{\frac{\overline{Y}}{\overline{\alpha}} - r}{(\sigma_V/\overline{\alpha})\sqrt{n}} = \frac{\hat{r} - r}{(\sigma_V/\overline{\alpha})\sqrt{n}}$$

A 100 (1 - δ)% confidence interval for $\bar{r}$ can now be formed as

$$P(\hat{r} - \frac{Z_t \sigma_V}{\bar{\alpha} \sqrt{n}} < r < \hat{r} + \frac{Z_T \sigma_V}{\bar{\alpha} \sqrt{n}}) \cong 1 - \delta/2$$

where

$$P(Z < Z_T) = 1 - \delta/2$$

with Z being a standard normal random variable

$$P(Z < Z_T) \overset{\Delta}{=} 1 - \text{erfc}\ (Z_T).$$

Unfortunately, σ_V is unknown thus we are forced to use its estimated values. Note that

$$\sigma_V{}^2 = E\{(Y - r\alpha)^2\}$$

$$= \text{var}\ \{Y\} + r^2\ \text{Var}\{\alpha\} - 2\ r\ \text{Cov}(Y, \alpha)$$

and Cov(Y, α) is not zero. The variance of Y and α and their covariances may be replaced by their estimates S_Y^2, S_α^2 and $S_{Y\alpha}$ then

$$S_V^2 = S_Y^2 + 2(\hat{r})^2\ S_\alpha^2 - 2\ \hat{r}\ S_{Y\alpha}$$

Therefore, the width of 100(1 - δ)% confidence interval for the performance measure r is

$$\frac{2\ Z_T S_V}{\bar{\alpha} \sqrt{n}}$$

It must be emphasized here that the regenerative method is only valid when the underlying queueing processes is regenerative. Further, the method will only be practical if many regenerative cycles can be observed, i.e., n large. In general, the length of the busy cycle cannot be analytically predicted, however, for an M/G/1 queue the average busy cycle length [4] is

$$E\{BP\} = \frac{E\{X\}}{1 - \rho}$$

where $E\{X\}$ is the average message length in seconds. Thus, we can expect for reasonable levels of traffic intensity, i.e., ρ not very close to 1, many busy cycles can be simulated. Unfortunately, in computer systems (not communications systems) performance evaluation, the regenerative method is not generally applicable since most processes arising in computer performance evaluation are either not regenerative or contain too few regeneration points to produce valid confidence intervals unless the run is extremely long. This is because computer systems commonly contain multiserver queues.

An alternate method for dealing with the correlation of the output sequence is to estimate the autocorrelation function. The estimated autocorrelation function is then used to estimate the variance of the sequence of packet delays and this variance is used to form a confidence interval. A single run method, called the spectral method, that estimates the correlation in the sequence has been used [5]. This method uses the fact that $\sigma^2(\hat{\mu}) \approx p(0)/(N - N_0)$ where p(0) is the spectral density of the process at zero frequency. The spectral density is the Fourier cosine transform of the covariance function $\{\gamma_k\}$ of the process defined by

$$p(f) = \sum_{k=-\infty}^{\infty} \gamma_k \cos(2\pi fk)$$

Spectral estimation techniques are applied to estimate p(0). This method has been shown empirically to produce satisfactory results for a variety of computer performance models and has been incorporated into a run length control procedures. Clearly direct autocorrelation function estimates can also be used.

Another method which has been applied is parametric modeling [5]. Here a parametric model is estimated which characterizes the output sequence. All time series analysis techniques are applicable. For example, Box-Jenkins methodology for model identification and estimation has been used. However, time series techniques have not produced reliable estimates of the variance of the sample mean [5]. The reason for this poor performance could be the nonstationary behavior of the time series and/or the non-normality of the observations.

In the previous two chapters we have discussed the statistical problems encountered in the simulation of data communications systems. It was recently noted [34] that a very common mode of operation for unknowledgeable simulation practitioners was to make a single simulation run of somewhat arbitrary length and then treat the resulting estimates as being the "true" characteristics of the model. Further, it appeared that if any output data analysis was being done then replication was practically the only approach ever used. The design of PCNSAS must take the statistical nature of the simulation into account. More importantly, it must unburden the network analyst with the statistical analysis whenever possible. The network analyst is not interested in determining how long the simulation needs to run, he simply wants an answer with a reasonable confidence interval. The need for small confidence intervals combined with the variance expansion due to the correlation in the output sequences implies that quite long simulation runs may need to be performed. Even with today's supercomputers, simulation of large complex networks is not possible. As concluded in [19], "a key challenge is to be able to devise computationally efficient techniques to simulate models of increasingly complex systems." In the next chapter, we will discuss some efficiency enhancing techniques which have been developed for queueing systems.

4.5 Computationally Efficient Techniques for Data Communications Network Simulation

The need for computationally efficient techniques for obtaining performance predictions of queueing systems has been known for some time [35]. All efficiency enhancing techniques are based on the proper exploitation of a-priori qualitative knowledge of the operating characteristics of the queueing system under study. In analytical models we have complete a-priori knowledge, however, as previously discussed, analytic models do not currently possess the required flexibility. Clearly, there is a trade-off between the degree to which a-priori knowledge is factored into a simulation model and its flexibility. In assessing the suitability of various efficiency enhancing techniques, this trade-off must be considered. Therefore, efficiency enhancing techniques will be evaluated in terms of their efficiency gained as a function of the flexibility lost.

Efficiency can be defined based on the percentage reduction in the volume of sampling required to attain a fixed confidence interval as compared to completely random sampling (brute force simulation). Alternately, efficiency can be defined based on the percentage reduction in computational effort (computer time) to attain a fixed confidence interval as compared to completely random sampling.

The two common approaches to developing efficiency enhancing techniques are 1) variance reduction techniques (VRT), and 2) hybrid modeling. Variance reduction techniques are based on using knowledge of the random processes involved in the system to obtain an estimate of the system performance. The random processes are manipulated, modified or combined to produce an alternate estimate which has the same expected value as the brute force prediction, however, with a smaller variance. Variance reduction techniques have been studied for many years [14], however their direct application to data communication network simulation has rarely been mentioned. An example of a VRT is modified Monte Carlo or importance sampling. This VRT has been applied with great success to performance analysis of communications links [40]. In this application of importance sampling the input random process is biased so that more bit errors are observed. The simulated biased bit error probabilities are then unbiased to obtain valid estimates of the bit error probability. Hybrid techniques are based on using a-priori knowledge in the form of an analytic submodel as part of a total simulation model (see [13] for a survey of hybrid modeling). Hybrid techniques have been applied to data communications network modeling [36, 37].

An example of a hybrid technique is the quasi-analytical estimation of bit error probability used in communication link simulation [42]. In this application of the hybrid techniques, simulation is used to estimate the detector output due to intersymbol interference, then an analytic model uses this estimate to obtain an estimate of bit error probability given the observed level of intersymbol interference. An average bit error probability is obtained by averaging over many levels of intersymbol interference.

The purpose of this section is to provide an overview of the state-of-the-art in efficiency enhancing techniques for network simulation and evaluate the existing results and approaches relative to PCNSAS. An overview of efficiency enhancing techniques will be presented first. This will be followed by examples of how such techniques have been applied.

4.5.1 An Overview of Efficiency Enhancing Techniques for System Simulation

In this section, several classical variance reduction techniques will be introduced. Variance reduction techniques can be developed because we have control over the generation and processing of all random processes involved in the simulation. This control or knowledge can be expolited to substantially increase the efficiency and effectiveness of any statistical simulation.

Three techniques which have evolved and proved useful in simulation will be discussed here. The importance sampling (modified Monte-Carlo) techniques are based on applying a distortion to the system input random variables. This distortion will give more weight to the values of the input which contribute the most to the expected value of the system performance measure. Control variates or regression sampling is based on obtaining a system variable which is correlated with the system performance measure. When appropriately applied (as discussed below), the covariance between the control variate and the performance measure can be used to obtain a variance reduction. The antithetic sampling technique also relies on a correlation, only with this technique different input random sequences are used which are known to produce correlated system responses. All these three techniques have been applied to the queueing system simulation problem with varying degrees of success [14, 38]. Several studies will be discussed in following sections.

4.5.1.1 Antithetic Sampling

Suppose that the sequence of pseudorandom numbers $\overline{X}_1 = (X_1, X_2, \cdots, X_m)$ are used to drive a simulation which produces a performance measure Y_1. Similarly, let $\overline{X}_2$ and Y_2 be stimulus and response for a second simulation of the same system. In general, we would form an average performance measure Y as

$$Y = \frac{Y_1 + Y_2}{2}$$

If Y_1 and Y_2 are statistically independent then

$$\sigma_Y^2 = \frac{\sigma_{Y_1}^2}{4} = \frac{\sigma_{Y_2}^2}{4}$$

However, if we select the random sequences $\overline{X}_1$ and $\overline{X}_2$ such that Y_1 and Y_2 are correlated then

$$\sigma_Y^2 = \frac{1}{4}\left(\sigma_{y_1}^2 + \sigma_{y_2}^2 + 2 \text{ cov } (y_1, y_2)\right)$$

If the variance due to the simulation is σ^2 (i.e., $\sigma_{Y_1}^2 = \sigma_{Y_2}^2 = \sigma^2$) then

$$\sigma_Y^2 = \frac{\sigma^2}{2} + \frac{1}{2} \text{ cov } (y_1, y_2)$$

Note that the correlation coefficient for Y_1, Y_2 is

$$\rho = \frac{\text{cov } (Y_1, Y_2)}{\sqrt{\sigma_{Y_1}^2 \sigma_{Y_2}^2}} = \frac{\text{cov } (Y_1, Y_2)}{\sigma^2}$$

Therefore

$$\sigma_Y^2 = \frac{\sigma^2}{2} + \frac{1}{2} \rho\sigma^2 = \frac{1}{2} \sigma^2 (1 + \rho)$$

Clearly, if the selected random number sequences $\overline{X}_1$ and $\overline{X}_2$ which produce responses which are negatively correlated ($\rho < 0$) then the variance of the performance measure has been reduced more than that expected from simply obtaining an additional sample. This is the concept behind antithetic sampling (for more details see [14]). The key to the application of antithetic sampling is knowledge that the system response is monotonically related to its stimulus. For example, consider two random sequences of uniform random numbers $\overline{U}_1 = (U_1, U_2, \cdots, U_m)$ and $\overline{U}_2 = (1 - U_1, 1 - U_2, 1 - U_3, \cdots 1 - U_m)$. If there is a monotic system relationship then the two system responses Y_1 and Y_2 generated from $\overline{U}_1$ and $\overline{U}_2$ would be negatively correlated and a variance reduction obtained. Application of antithetic sampling to queueing systems was considered by Fishman in [39].

4.5.1.2 Importance Sampling

The concept of importance sampling is based on generalized expected value, i.e., given $Y = g(X)$ then

$$E\{Y\} = \int_{-\infty}^{\infty} g(x)\, f_X(x)\, dx$$

where

$$f_X(x) = \text{pdf for the random variable } X$$

Consider the two systems shown in Figure 4.5.1.

Here let $f_X(\cdot)$ and $f_W(\cdot)$ be the known pdf's for the input random variables X and W respectively. The system $g(\cdot)$ represents the mapping of the input random variables to the output performance measure accomplished by the simulation. That is, Y is the system performance with a stimulus X while Z is the output with the stimulus W.

Using the generalized expected value we find that

$$E\{Y\} = \int g(x)\, f_X(x)\, dx$$

and

$$E\{Y_1\} = \int g(w) \frac{f_X(w)}{f_w(w)} f_w(w)\, dw = E\{Y\}$$

Therefore, the system shown in Figure 4.5.2b can be used to estimate the performance measure Y from the simulation. The variance of Y_1, the modified performance measure is given by

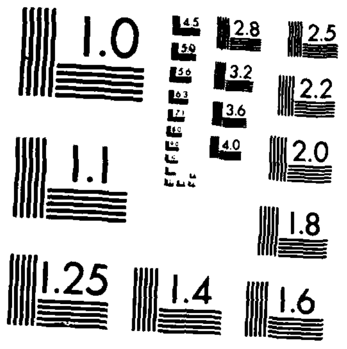
$$\sigma_{Y_1}^2 = \int_{-\infty}^{\infty} \left[\frac{g(w)\, f_X(w)}{f_w(w)}\right]^2 f_w(w)\, dw - (E\{Y\})^2$$

If we could set

$$\frac{g(w)\, f_X(w)}{f_w(w)} = E\{Y\}$$

AD-A174 316   A PACKET COMMUNICATION NETWORK SYNTHESIS AND ANALYSIS   2/2
SYSTEM(U) STAR CORP LAWRENCE KS K S SHANMUGAN ET AL
AUG 86 STAR-0112-1 RADC-TR-86-139 F30602-85-C-0112

UNCLASSIFIED   F/G 17/2   NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

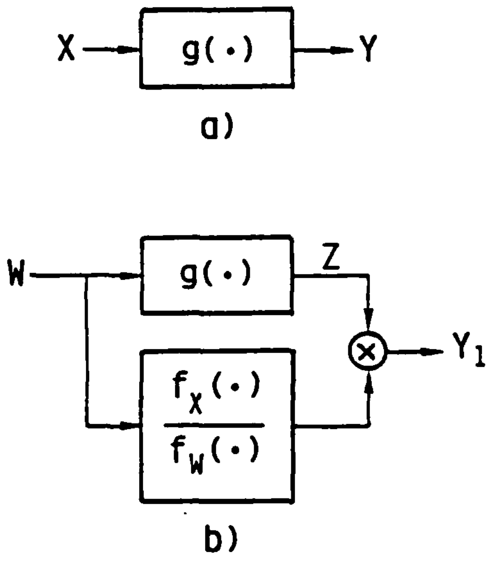


Figure 4.5.1 Block Diagram Model for Importance Sampling

then $\sigma_{y_1}^2$ could be reduced to zero. Clearly, we cannot achieve this reduction because we do not know $E\{Y\}$, the value we are trying to estimate using the simulation. A problem in importance sampling is given $f_X(x)$ and some knowledge of $g(\cdot)$ find a suitable function $f_w(w)$ which yields a variance reduction. Shanmugan [40] suggests the following form for $f_w(\cdot)$ for the case when $f_X(x)$ is a zero mean Gaussian

$$f_w(w) = \frac{cf_X(w)}{[f_X(w)]^{\alpha}}$$

where α and c are selected such that $f_w(w)$ is a pdf.

While Mitchell [41] suggests

$$\begin{aligned} f_w(w) &= \frac{1}{\mu_m} e^{-w/\mu_m} && w > 0 \\ &= 0 && w < 0 \end{aligned}$$

when X is exponential with mean μ.

Note in both of these cases the biased (W) and unbiased (X) pdf's have the same form. This represents a convenience in that the same random number generator may be used.

The above example considered a system that was memoryless. Shanmugan [40] shows that importance sampling can also be applied to systems with memory. For details of the procedure see [40]. Importance sampling was applied to queueing systems by Moy [38]. For a simple single server queue, Moy obtained about 50% variance reduction, however, for a more complex (multi-server) system, the variance actually increased unless complicated forms of biasing are used.

4.5.1.3 Regression Sampling

Regression sampling (or control variates) is a variance reduction technique which also relies on the correlation between simulation random variates. Let C be a random variable which is produced during the simulation. Assume that the mean value of C can be analytically predicted. Let Y denote the performance measure obtained from the simulation. Then form another variate Y_R as

$$Y_R = Y - \alpha(C - \mu_C)$$

where

$$\alpha = \text{constant}$$
$$\mu_C = E\{C\}$$

Clearly

$$E\{Y_R\} = E\{Y\}$$

so the variate Y_R can be used to estimate the system performance measure. The variance of Y_R is given by

$$\sigma_{Y_R}^2 = \sigma_Y^2 + \alpha^2 \sigma_C^2 - 2\alpha \operatorname{Cov}\{Y, C\}$$

This variance is minimized if

$$\alpha = \frac{\operatorname{Cov}\{Y, C\}}{\operatorname{Var}\{C\}} .$$

Setting α to its optimum value then

$$\sigma_{Y_R}^2 = (1 - \rho_{Y_C}^2)\, \sigma_Y^2$$

Thus, if a random variate within the simulation can be found which has a high positive correlation with Y and its expected value is known then a significant variance reduction can be obtained. Regression sampling has been combined with regenerative techniques in several studies [27, 43, 44, 45] as will be discussed in detail later.

4.5.1.4 Hybrid Techniques for Queueing System Simulation

In hybrid simulation, some knowledge of the system operation in the form of an analytic submodel is incorporated into the performance prediction mechanism. Clearly, in hybrid simulation we are trading off flexibility for efficiency. The analytic submodel is always based on a set of assumptions which restrict its general applicability. The value of hybrid simulation can be theoretically established by the Rao-Blackwell Theorem [46]. Here, assume prior knowledge of a function $\phi(\cdot)$ such that

$$E\{\phi(X)\} = \mu$$

where μ is the expected value of the desired performance measure. Further we know that

$$E\{Y|X = x\} = \phi(x)$$

and $E\{Y\} = \mu$ where Y is the random variable modeling the performance measure. The Rao-Blackwell Theorem states that

$$\sigma^2_{\phi(x)} < \sigma^2_Y$$

That is, use of the prior knowledge $\phi(X)$ only can help in the estimation of μ. In general, hybrid techniques offer a significant advantage. However, they not only reduce the simulation flexibility but require a high level of expertise on the part of the analyst. An overview of hybrid techniques is given in [13]. Several specific studies [36, 37] will be summarized in the following sections.

4.5.2 A Survey of Hybrid Techniques for Communications Network Simulation

Four cases where hybrid techniques have been applied to queueing system simulation will be reviewed here relative to their applicability to PCNSAS.

4.5.2.1 Simulation of Aggregate Background Traffic for Simulation of a CSMA/CD LAN.

The computational requirements for a simulation of a network increases as the number of nodes increases. O'Reilly and Hammond [37] were faced with the problem of modeling a CSMA/CD network with 10,000 nodes. Clearly, brute force simulation was not feasible. To overcome this difficulty they realized that for a broadcast network it would be possible to partition the nodes into two groups: primary nodes which are simulated in detail and background nodes. Background nodes are modeled as one source which based on an analytic model alternates between busy and idle states. (For details of the analytic model see [37]). As usual, a set of assumptions were made in the development of the analytic model, the most significant ones being:

1. All background nodes have Poisson interarrival process.

2. All background nodes are identical.

Figure 4.5.2 shows a comparison in computational efficiency between the method derived in [37] and brute force technique. Also shown is the partition method when rare states are removed from the analytic model. To further explain this truncation process, define p_i as the probability of i new arrivals attempting transmission in a typical time slot. (A time slot is usually the maximum one-way propagation delay between stations on the network). Define q_i as the probability of i attempted retransmissions occuring during a time slot. These basic probabilities govern the behavior of the system and are analytically calculated in [37]. It is claimed in [37] that, under steady-state conditions, the probability of having more than a small number of arrivals to the network, from either the backlogged (retransmissions) or thinking (new arrivals) stations is extremely small. This is the justification for limiting i (the number of arrivals) and thus the number of probabilities which must be calculated. Unfortunately, the accuracy of this truncation procedure is not quantatively addressed in [37]. It is simply stated that in practice, truncation of the number of p_i and q_i probabilities to be evaluated would always seem possible while maintaining the required accuracy. We can conclude from this figure that the partition techniques with rare

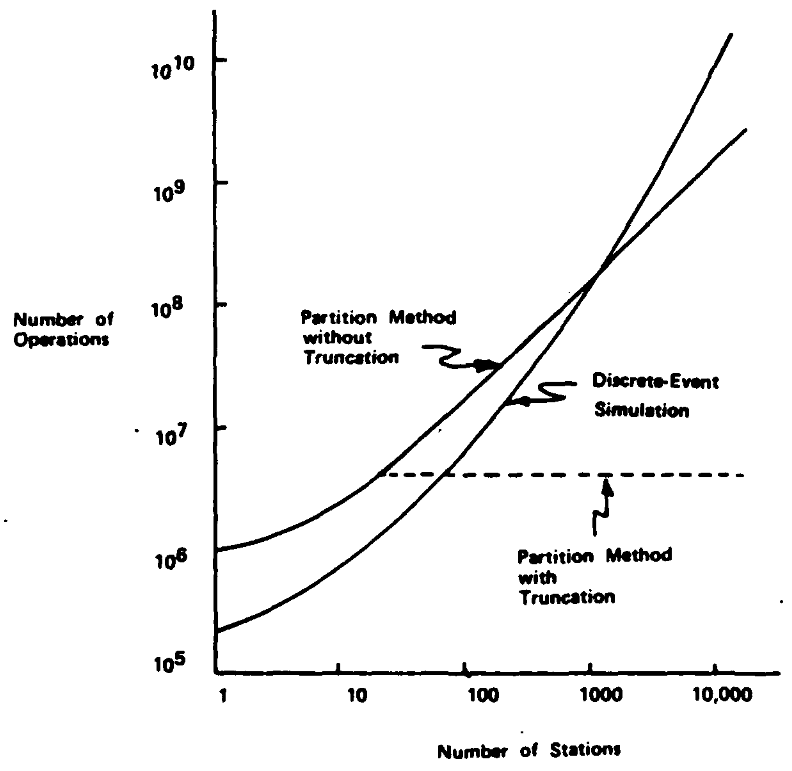


Figure 4.5.2 Run time comparison between the partition method and Brute Force simulation for constant load of 100% (from [30])

event truncation is significantly more efficient than the brute force approach for large numbers of nodes. Further, this result illustrates that the added computational load of the analytic model must be considered. Here for networks with fewer than 100 nodes the brute force techniques is superior. As in all hybrid techniques, the assumptions on the background nodes may make this approach too restrictive for some applications.

4.5.2.2 Application of Aggregate Background Traffic Modeling to PCNSAS.

The study of O'Reilly and Hammond [37] clearly demonstrates the potential of using an aggregate background traffic method to improve the efficiency of broadcast radio networks. We feel that a state dependent Markov model could be developed to model background traffic in a packet radio network. Such a model would allow for the performance evaluation of large packet radio networks. However, a significant effort would be required to develop a model which not only improves the computational efficiency of the simulation, but is still flexible enough to be useful in obtaining performance data on networks of interest.

4.5.2.3 Hierarchical Modeling of Local Area Computer Networks

In hierarchical modeling, subnetworks are defined and described by some simpler network or if possible, by a single server queue. Wong, et al., [36] have applied this principle to the modeling of a token LAN used for a file transfer application. The technique used here decomposed the complete network into three levels: Level 1 models the transmission protocol, Level 2 models a virtual circuit with a window protocol and Level 3 a file transfer application. Figure 4.5.3 summarizes the approach. An analytic model is used to predict the Level 1 performance. Here the performance is the mean service time for the ℓth node data and receive ready packets, and the additional delay for all other nodes from the ℓth node's traffic. This information is supplied to a simulation model for Level 2. The Level 2 simulation model predicts the mean time to transfer a file given k transfers in progress. The Level 3 model predicts the mean response time given N nodes on the network using an analytic model and the Level 2 results.

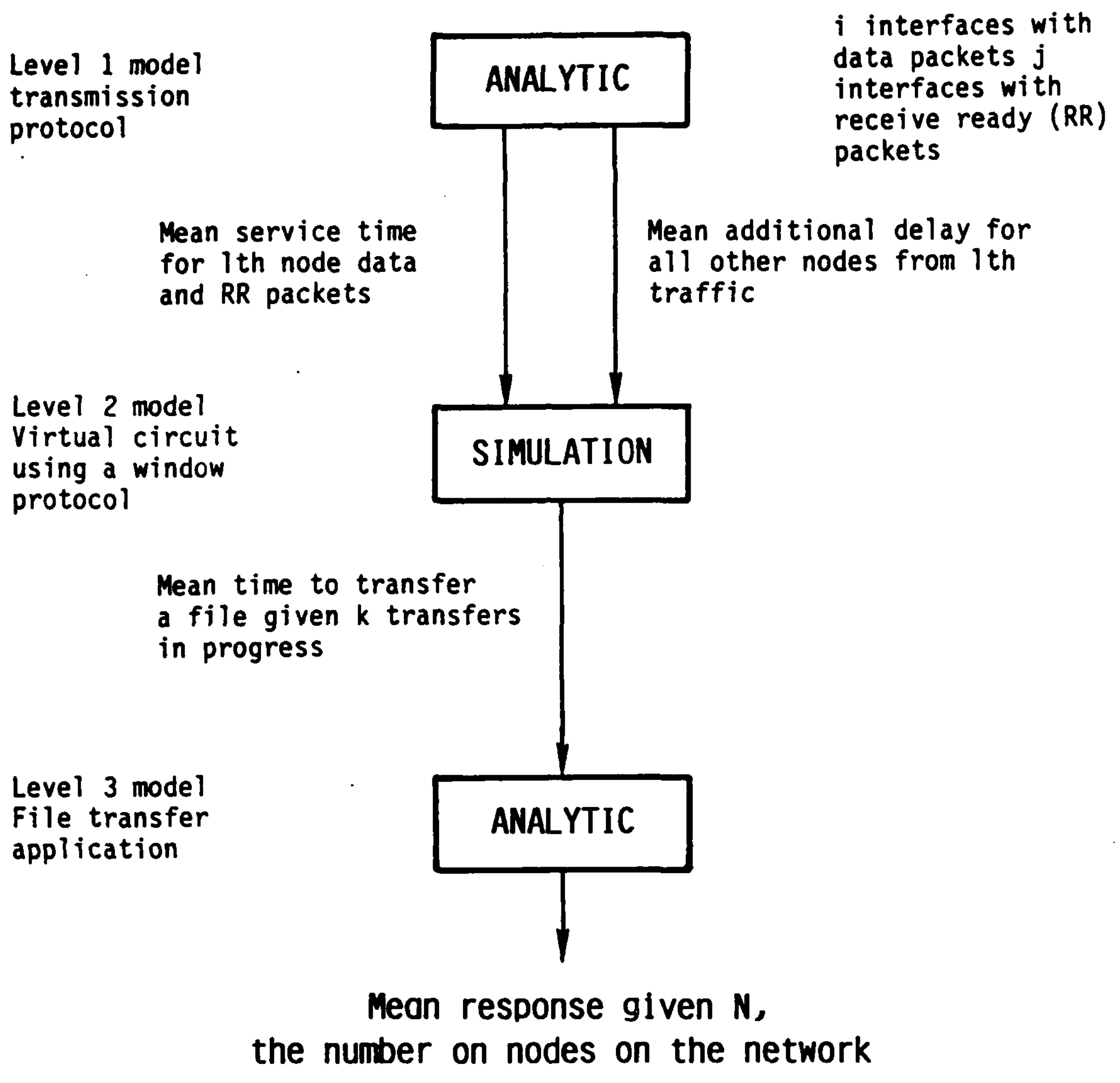

Level 1 model transmission protocol
ANALYTIC
i interfaces with data packets j interfaces with receive ready (RR) packets
Mean service time for lth node data and RR packets
Mean additional delay for all other nodes from lth traffic
Level 2 model Virtual circuit using a window protocol
SIMULATION
Mean time to transfer a file given k transfers in progress
Level 3 model File transfer application
ANALYTIC
Mean response given N, the number on nodes on the network

Figure 4.5.3 Hierarchical modeling of a local area computer network token ring with a file transfer application

An efficiency improvement factor of 50 over brute force simulation was obtained. However, very restrictive simplifying assumptions were required to allow for the interface of the models for the three levels. For example, no call set up was modeled, no transmission errors were allowed, a zero token passing time was assumed and a Markov chain analysis was used.

Hierarchical techniques show great promise for improving the efficiency of modeling specific networks. However, PCNSAS is intended to be a flexible tool, and as illustrated by this example, hierarchical techniques do not presently allow much flexibility. Incorporating hierarchical modeling into PCNSAS would be desirable if it would not restrict its flexibilty.

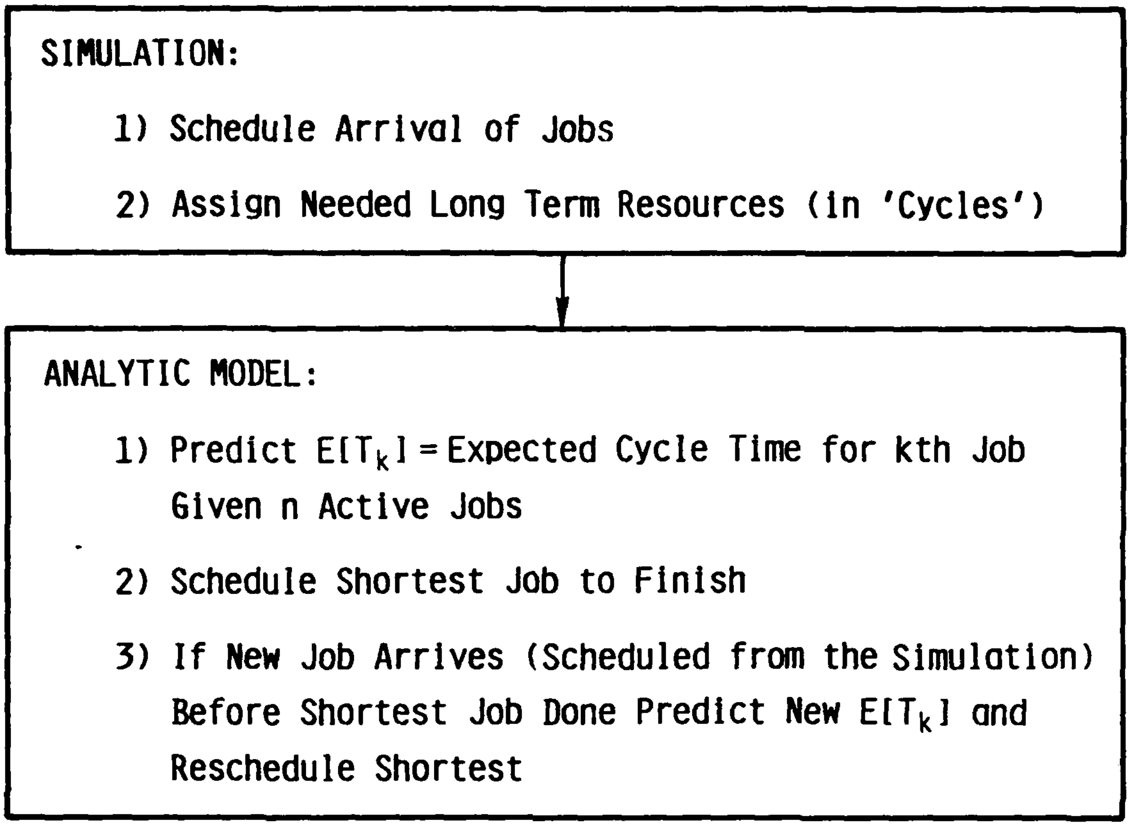
4.5.2.4 Hybrid Modeling of Computer Systems

Hybrid modeling has been successfully applied to the simulation of computer systems [47, 48, 49]. These techniques are based on the decomposition of models into low and high frequency events. Low frequency events could be the arrival and scheduling of jobs in the system. High frequency events could be the use of short term resources, e.g., CPU, I/O processors and I/O devices. The low frequency events are modeled using simulation while the high frequency events are represented in an analytic model. The process is illustrated in Figure 4.5.4. In this figure, simulation is shown to be used for scheduling the arrival of jobs and assigning the needed long term resources needed to process the job. The analytic model is used to predict the expected time to complete all jobs currently in the system. The analytic model then schedules the next job to finish. If a new job arrives before the next job completes, then the analytic model is used to reschedule the next job to finish. For this type of hybrid model to work it is required that there be no interaction between the high and low frequency events. Unfortunately, this independence does not exist in communication networks.

4.5.2.5 Combined Analytic/Simulation Model for G/G/1 Queues

The average waiting time for a G/G/1 queue has been found to be

$$E\{W\} = -\frac{E\{U^2\}}{2E\{U\}} - \frac{E\{I^2\}}{2E\{I\}}$$



SIMULATION:

1) Schedule Arrival of Jobs

2) Assign Needed Long Term Resources (in 'Cycles')

ANALYTIC MODEL:

1) Predict $E[T_k]$ = Expected Cycle Time for kth Job Given n Active Jobs

2) Schedule Shortest Job to Finish

3) If New Job Arrives (Scheduled from the Simulation) Before Shortest Job Done Predict New $E[T_k]$ and Reschedule Shortest

Figure 4.5.4 Representation of Hybrid Simulation of Computer Systems

where

W = waiting time

I = Duration of the idle period

$U_n = X_n - T_{n-1}$

X_n = service time for nth packet

T_{n-1} = interarrival time between n-1 and nth packet

Note that on average, U_n will be negative so that $E\{W\}$ will be positive. Given the arrival processes, the moments of U can be analytically found. Unfortunately, in general the moments for the idle period cannot be analytically predicted. Recently Minh and Sori [50] have proposed using a simulation to find the moments of I and then using the above equation to find the average waiting time. It is claimed that the variance of this hybrid estimation is substantially reduced at high traffic intensities. This combined analytic/-simulation is simple and flexible. However, further research is needed to determine its applicability to networks of queues, to quantatively determine the variance reduction obtained by using it, and its role in PCNSAS. Of all the combined analytic/simulation techniques considered here, this one has the most immediate potential for being useful in PCNSAS.

4.5.2.6 Application of Perturbation Analysis to Network Simulation

A recently developed technique called perturbation analysis (PA) was introduced in 1983 by Ho and Cao [53] and very recently discussed in terms of communications networks by Cassandras [54]. Peturbation analysis is claimed to combine the advantages of both theoretical and simulation approaches, while minimizing their disadvantages. Perturbation analysis provides information on the sensitivity of network throughput relative to various system parameters. This information can be used for the optimization of queueing networks. The purpose of this section is to present the basic idea of perturbation analysis, present its assumptions, and discuss its potential for PCNSAS.

As in [53] we will consider a tandem network of finite queues to illustrate the basic idea of perturbation analysis. Every queue in the tandem network can be in one of three states. The queue can be serving a message and is thus busy or can be idle because previous (upstream) queues are empty (this called the no input (NI) state), or can be blocked because downstream queues are full (this is called full output (FO) state). Perturbation analysis is based on the premise that the FO and NI intervals are the only means of interaction among the servers. Each queue can then be represented by a sequence of events, where each event duration is of some random length.

Let E_i denote this sequence for the i^{th} queue or server and $S_i(j)$ be service time for the i^{th} server on activation j as shown in Figure 4.5.5. Peturbation analysis is based on the idea of introducing an "infinitesimal" perturbation in E_i at time t, e.g., the j^{th} activation is initiated some Δ early. Such a perturbation will propagate to other event sequences E_k $(k \neq i)$ only through the FO and NI intervals. Because a perturbation can be propagated only through FO or NI intervals, not all induced perturbations are propagated. A perturbation is said to be realized if it is propagated to all other servers otherwise it is said to be lost. Once a perturbation is realized it is equivalent to advancing or delaying the entire event sequence by the amount of the perturbation.

We are most interested in perturbing a parameter of the network, e.g., mean service time (packet length) rather than specific activation. In this case, a parameter perturbation will induce a series of perturbations which are propagated according to the perturbation analysis rules [53, 54]. The percentage of induced perturbations realized is called the realization ratio. The realization ratio is then used to calculate an estimate of the derivative of the throughput relative to the parameter (see [53] for a justification and example). That is, the sensitivity of the throughput to a parameter changes. As discussed in [54], we can see that perturbation analysis consists of three steps: 1) perturbation generation; 2) perturbation propagation; and 3) performance measurement.

Now that the concept of perturbation analysis has been introduced, we can relate it to communication network simulation. First note that perturbation analysis relies on the existence of a nominal (sample) realization similar to that shown in Figure 4.5.5. Such a realization can be obtained through measurements on the network or via simulation. To this extent, perturbation analysis has all the advantages of the simulation approach to network analysis. Using perturbation analysis we obtain the added benefit of using one

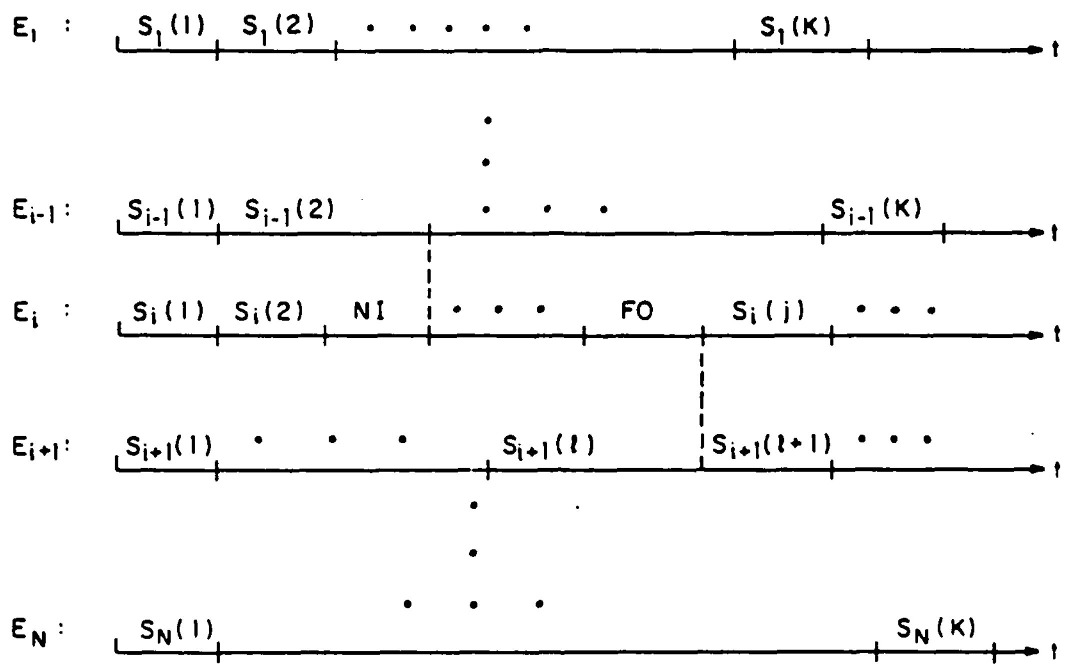


Figure 4.5.5 Representation of operating history in a tandem network (from [53])

sample realization combined with analysis to derive answers to the question: "What will happen if we repeat the simulation (sample realization) exactly except for a small perturbation at some time t?" In fact, perturbation analysis allows for the solution of multiple what/if questions simultaneously based on one sample realization. This is where the efficiency of perturbation analysis arises.

The concept of perturbation analysis is straight forward however, the application of the approach is not. The accuracy of the approach is seriously degraded if one major assumption is violated. This assumption is that the perturbations are sufficiently small so that the order of events in the sample realization remain unchanged at each queue. Perturbations can only change the duration of the state not the order of the states in the event sequence. This assumption currently limits the applicability of perturbation analysis because large perturbations are of more interest. In some networks, e.g., CSMA/CD or broadcast networks, small perturbations can easily change the order of events. Current work [54] is progressing in this area as is perturbation analysis algorithms for dynamic routing.

Perturbation analysis is an exciting new area for network performance analysis. Its potential for direct optimization is very interesting. However, further research is needed to overcome its limitations before it can be applied in a general way to PCNSAS.

4.5.2.7 Usefulness of Hybrid Techniques for PCNSAS

Hybrid techniques can be effective in reducing the computational requirements for network simulation. However, these techniques in general require extensive customization. They must be specifically designed for a particular network. It will be difficult to develop hybrid techniques (except for the one discussed in Section 4.5.2.5) which can be easily modified to accommodate different systems. Further, each technique would greatly increase the difficulty in extending and adding new features to PCNSAS. Additional work needs to be done before hybrid techniques can be generally applied to communication network simulation.

4.5.3 A Survey of Variance Reduction Techniques for Communications Network Simulation

Several variance reduction techniques have been applied to the simulation of queueing systems. The purpose of this section is to survey these techniques and discuss their merits relative to PCNSAS. Most of these techniques rely on the regenerative nature of the queueing systems. Remember that variance reduction techniques are based on modification of the simulation random processes.

4.5.3.1 Efficient Estimates for Simulated Queueing Systems

It is well known that the performance measures for a queue are related. For example, Little's formula relates delay and average number in the queue. The question then is which measure do you estimate in the simulation? A. M. Law has conducted a study [51] on efficient estimators. For single server queues Law concluded that

$$\hat{d} = \frac{E\{X\}\ \bar{D}_c}{\bar{B}}$$

where

$\hat{d}$ = estimate for average delay

X = service time

$\bar{D}_c$ = average total delay of all packets in a busy cycle

$\bar{B}$ = average length of the busy cycle

was the most efficient estimator for the average delay.

In addition to its direct application, this study shows that the post-processor for PCNSAS must be designed to handle a variety of different measurements.

4.5.3.2 Regenerative Simulation with Controls Variables

The concepts of regenerative simulation and control variables were combined by several authors to form a new variance reduction technique for queueing systems. The major problem in applying control variables is the identification of suitable control variables. The problem here is finding variables which are highly correlated with the performance measure and whose expected value can be analytically predicted. In [43] several different control variables were proposed. These control variables were defined over each busy cycle. That is, the control variables were internal to a cycle. A variance reduction of 50% was obtained for a M/M/1 queue. Further, it was found that it was not possible to analytically predict the expected percent variance reduction. It was also found that the percent variance reduction was reduced by increasing the traffic intensity. This observation is not encouraging because it is at these high loads where the most variance reduction is needed.

This approach was extended to networks of queues by Lavenberg, et al., [44]. Again, a class of control variables were proposed for closed queueing networks and experiments conducted to evaluate their effectiveness (see [44] for details). The loss in variance reduction due to estimating the optimum regression parameter, α (see Section 5.1.3) was quantified. Percentage variance reductions in the range of 15 to 70% were obtained. The main result relative to PCNSAS is that the technique can be applied to networks of queues.

To overcome part of the problem in finding suitable control variables Heidelberger [27] proposed to use several different estimates for the performance parameters and then use a minimum variance combination to form his final estimate. His approach is restricted to Markov processes. Again, the percent variance reduction decreased as the traffic intensity increased.

The concepts of regeneration, and antithetic sampling were combined by Fishman [39] to obtain a variance reduction. Fishman's technique makes the results from various cycles negatively correlated. For a M/M/1 queue, the variance reduced by a factor of 13. More importantly, the percentage variance reduction did not decrease as the traffic intensity increased. However, the technique is limited to Markov models.

4.5.3.3 A Variance Reduction Technique Based on Conditional Expectations: Another Look at Hybrid Techniques

As discussed in [55] a variance reduction technique based on conditioning is aimed at exploiting some special property of a model's structure so that we can sometimes replace an estimate of a quantity by its true, theoretical value. In removing this source of random fluctuation, we hope that the variance of the final output random variable will be reduced, although there can be no absolute guarantee of a variance reduction; again, pilot studies comparing the conditioning technique with a straight forward sampling approach could indicate whether (and to what extent) the estimated variance is being reduced. Since this variance reduction technique is highly model-dependent, we cannot give any general discussion of how it can be accomplished or how effective it may be.

Suppose that Z is an output random variable of interest (such as a delay in queue of a message) and we seek to estimate $\mu = E(Z)$. Suppose that there is some other random variable Y such that we can calculate (exactly) the conditional expectation $E(Z|Y = y)$ for any real number y. (Note that $E(Z|Y = y)$ is a function of the real number y, whereas $E(Z|Y)$ is a random variable). Then $\mu = E(Z) = E[E(Z|Y)]$, so that the random variable $E(Z|Y)$ is unbiased for μ. Further,

$$\mathrm{Var}[E(Z|Y)] = \mathrm{Var}(Z) - E[\mathrm{Var}(Z|Y)] \leq \mathrm{Var}(Z)$$

indicating that observing $E(Z|Y)$ (as computed analytically from the random variable Y) instead of direct observation of Z will result in a smaller variance. The trick, of course, is specifying a Y so that $E(Z|Y)$ can be computed (one would hope efficiently) and so that $E[\mathrm{Var}(Z|Y)]$ is large. The situation is further complicated by the fact that the above equation will not generally be applicable to the final output random variables (such as an average of messages delays) due to autocorrelation in the simulation output, so that a variance reduction is not guaranteed.

Since this variance reduction technique is so heavily model-dependent, we illustrate it by describing a successful implementation which has been reported in the literature. The following example is taken from Carter and Ignall [52].

A simulation model was developed to compare alternative policies for dispatching fire trucks to fires in the Bronx. Certain fires are classified as "serious," since there is considerable danger that lives will be lost and property damage will be high unless the fi-- department is able to respond quickly with enough fire trucks. The goal of the simulation was to estimate the expected response time to a serious fire under a given dispatching policy.

Historical data indicated that about one of every thirty fires is serious. Thus, the model would have to progress through about thirty simulated fires to get a single observation on the response time to a serious fire, which could lead to very long and expensive runs to generate enough serious fires to get a good estimate of the desired expected response time. However, the model's specific structure was such that, given the state of the system (the location of all fire trucks) at any instant, the true expected response time to a serious fire could be calculated, should one occur at that instant. Furthermore, the probabilistic assumptions (in particular, serious fires were assumed to occur in accordance with a Poisson process) enabled the simulators to condition on the event of a serious fire's occurring at every instant when the system state was observed regardless of whether or not a serious fire actually did occur, without introducing bias. Thus, the simulation was simply interrupted periodically to observe the system state, and the expected response time to a serious fire, given the current state, was calculated and recorded, and the simulation was resumed. The final estimator was the average of these conditional expected response times and included many more samples than the number of serious fires actually simulated.

Carter and Ignall [52] found that the conditional-expectation approach reduced the estimated variance of their estimator by about 95 percent in comparison with the straight forward estimator. Computationally, however, the conditional expectation approach was more expensive, so that for the same computing cost the variance reduction was about 92 percent.

Clearly, it was the model's specific structure which allowed an analytic evaluation of the response time given the current state. Note that using their approach, the effect of the "rare" event could be calculated often, that is, at each state. Carter and Ignall introduce the concept of a rare event and show that when combined with the conditional expectation, a significant variance reduction can be achieved.

4.5.3.4 A New Variance Reduction Technique Based on Regenerative Processes and Rare Events

As with the development of alternative dispatching policies for fire trucks, computer networks also experience "serious" rare events which dominate the network performance. For example, the sudden degradation or total loss of a communications link or the transfer of a large high priority data file would dominate the network characteristics. Unfortunately, the complex structure of these networks preclude the development of a model which would allow the use of the conditional expectation or hybrid approach.

However, we have found that the concept of a rare event can be combined with the regenerative nature of queues in the network to develop a new variance reduction technique. To explain this new technique, we will consider a specific simple example. M/G/1 queue is shown in Figure 4.5.6. Here there are two sources of messages (or packets) for the queue. One source generates short (type 1) messages (.6 sec/message on average) very often (1 message/-sec), the second source generates very long (type 2) messages (1000 sec/message on average) but rarely (1 message every 10,000 sec). Both sources have exponential interarrival and message length distributions. It is easy to analytically calculate the average delay once the arrival and message length processes have been discussed. To characterize the arrival and message length processes, note that the merging of two independent Poisson processes is also a Poisson process [4] with a total arrival rate, λ_T, the sum of the input rates, here

$$\lambda_T = 1 + 10^{-4}. \quad \text{messages/sec}$$

The combined message length distribution, however, is no longer exponential. This distribution can be written as

$$f_X(x) = P_1 \; f_X(x|M_1) + P_2 \; f_X(x|M_2)$$

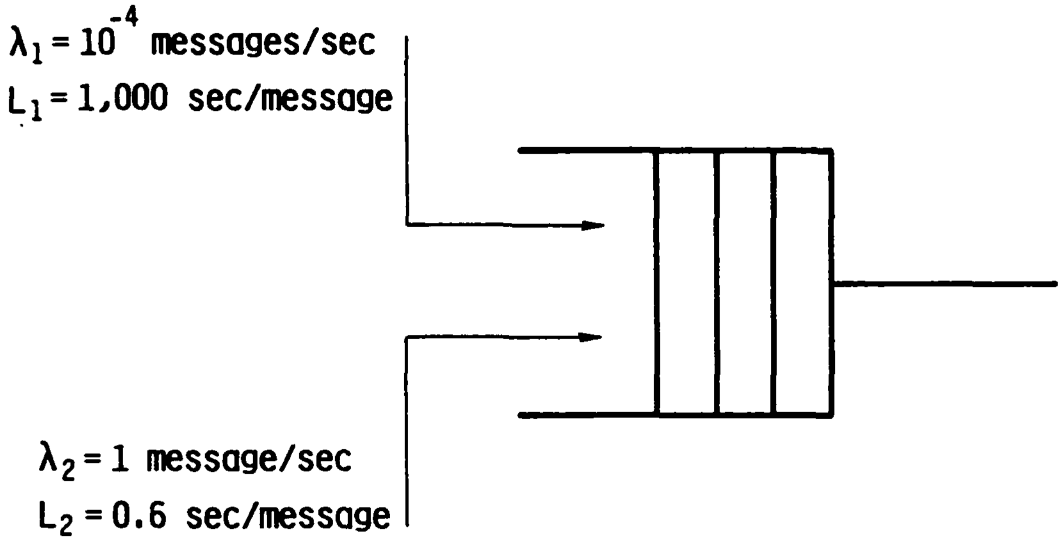

$\lambda_1 = 10^{-4}$ messages/sec
$L_1 = 1,000$ sec/message
$\lambda_2 = 1$ message/sec
$L_2 = 0.6$ sec/message

Figure 4.5.6 Example M/G/l Queue

where

P_1	= Probability that the message is of type 1
$f_X(x\|M_1)$	= length pdf given type 1 message
P_2	= Probability that the message is of type 2
$f_X(x\|M_2)$	= length pdf given type 2 message

In this example

$$P_1 = \frac{1}{\lambda_T} ,$$

and

$$P_2 = \frac{10^{-4}}{\lambda_T} .$$

As noted in Section 4.2.3, only the first two moments of $f_X(x)$ are needed to calculate the average delay for M/G/1 queues. In this case

$$E\{X\} = P_1 \; E\{X|M_1\} + P_2 \; E\{X|M_2\}$$

$$= .6999$$

and

$$E\{X^2\} = P_1 \; E\{X^2|M_1\} + P_2 \; E\{X^2|M_2\}$$

$$= 2 \; P_1 \; (E\{X|M_1\})^2 + 2 \; P_2(E\{X|M_2\})^2$$

$$= 200.699$$

and from [23] the average delay is found by

$$E\{T\} = \left(\frac{\lambda_T}{2} \right) \left(\frac{E\{X^2\}}{(1 - \rho_T)} \right) + E\{X\}$$

$$= 335.2$$

Clearly, there is no need to resort to simulation for the analysis of this queue, however, this simple case allows for the illustration of our new technique.

Figure 4.5.7 is a rough sketch of how the number of messages in the queue would vary with time for the M/G/1 queue considered above. Here a rare event is the arrival of a large packet. From this figure, we can draw several important conclusions. First, busy cycles can be divided into two basic classes. Class 1 busy cycles containing no rare events and thus look statistically identical to busy cycles from an M/M/1 queue for this case. Class 2 busy cycles contain one or more rare events. Second, because rare events are rare, we expect to observe many more class 1 busy cycles than type 2.

Now, consider how the regenerative simulation approach should be directly used (brute force) to calculate the average delay for this case. Note class 2 busy cycles can be further divided as to the exact number of rare events contained in the busy cycle. Let R be a positive integer random variable representing the number of rare events in a busy cycle. Then class 1 busy cycles are those where R = 0. Each transmitted packet can be associated with a busy cycle where R=k. That is, packets can be grouped into sets where membership in a set is determined by the properties of the busy cycle in which the packet was serviced. For example, we can construct a set of packets associated with each busy cycle where exactly k rare event occurred. That is, the sample space of all packets can be divided into subsets based on an associated busy cycle with R=k. Now, the average packet delay can be conditioned on its associated busy cycle. That is, define

$$E\{\text{Delay} | \text{the packet is a member of the subset of packets associated with busy cycle with } R = k\}$$

$$\triangleq E\{T | R = k\}$$

The unconditional average packet delay can now be defined as

$$E\{T\} = \sum_{k=0}^{\infty} E\{T|R=k\} \; P(R=k)$$

where P(R=k) is the probability that a packet was associated with the busy cycle R=k.

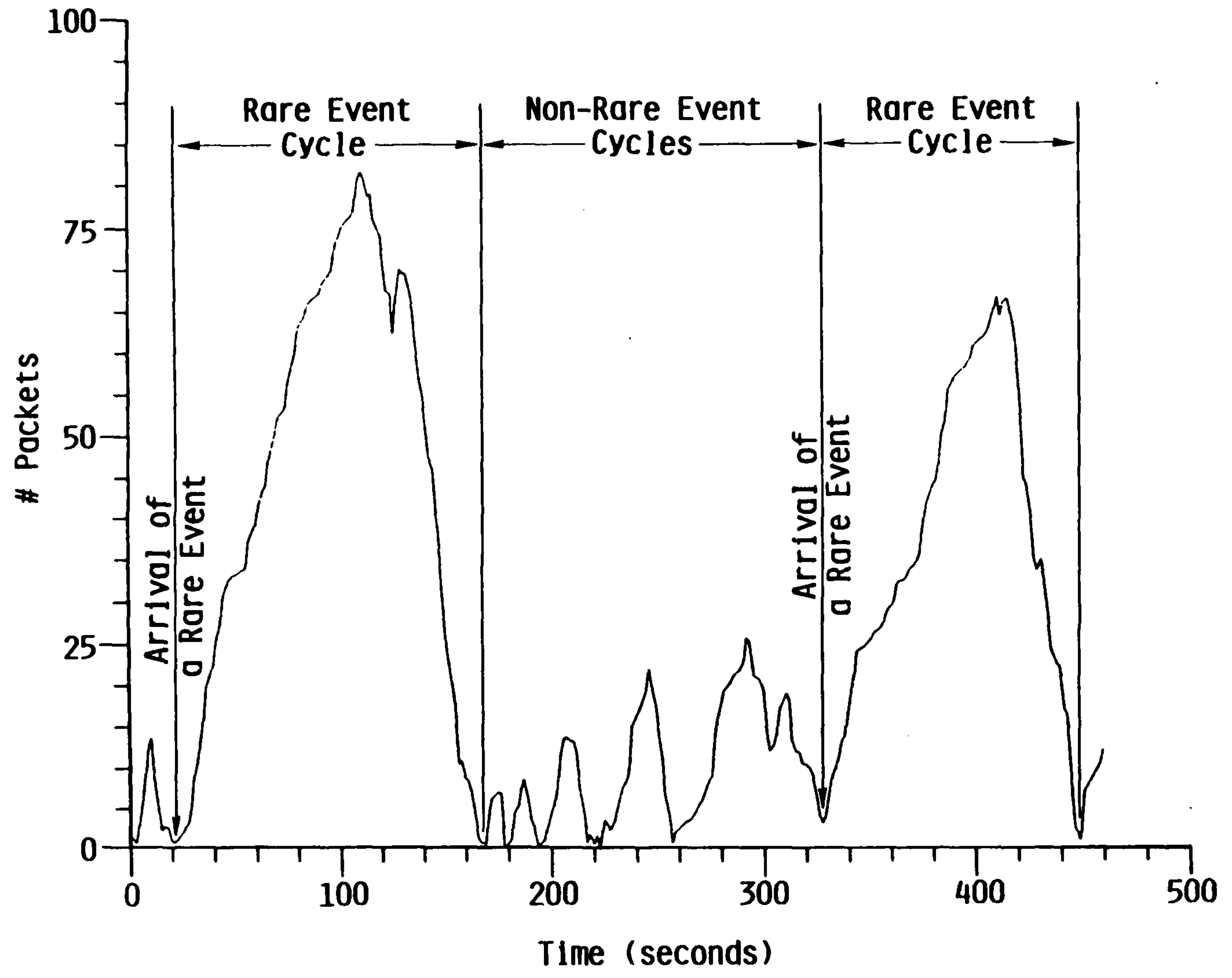


Figure 4.5.7 Typical time history for number in M/G/1 queue with rare events.

One way to directly apply (there is no variance reduction obtained here) the regenerative approach to this case is to measure the total number of packets occurring with R=k, $N|R=k$, and the total delay of all packets with R=k, $T_T|R=k$, then form

$$E\{T|R=k\} = \frac{E\{T_T|R=k\}}{E\{N|R=k\}}$$

To find the unconditional average delay we must estimate the P(R=k). This can be done quite easily by estimating P(R=k) as

$$\hat{P}(R=k) = \frac{\text{number of packets associated with } R = k}{\text{total number of packets processed}} = \frac{N|R=k}{N_T}$$

where N_T is the total number of packets processed in the simulation. A typical simulation result for this case is shown in Table 4.5.1.

The unconditional average delay can now be estimated from the regenerative simulation results by

$$\hat{T} = \sum_{k=0}^{\infty} \left[\frac{T_T|R=k}{N|R=k} \right] \left[\frac{N|R=k}{N_T} \right]$$

Clearly, this expression also can be written as

$$\hat{T} = \frac{\text{sum of all packet delays}}{\text{total number of packets processed}} = \frac{\sum_{k=0}^{\infty} T_T|R=k}{N_T}$$

However, the first expression will be useful in explaining how the new variance reduction technique is derived. Applying either of the above equations $\hat{T}$ for the one typical run shown is approximately 278 sec.*

In the first row of Table 4.5.1, we see that there were approximately 300,000 busy cycles with no rare events. These busy cycles would be indistinguishable from busy cycles from a M/M/1 queue. Further, note that about 75% of the packets processed were with no rare events. Clearly, if we could predict the delay without having to process the busy cycles with no rare

* As will be discussed later, this result is reasonable for one run given the variance of T.

events a significant saving can be obtained. This is the basis for the new variance reduction technique.

First consider how we might remove all the non-rare event cycles from the simulation. Suppose we begin the simulation with the arrival of a rare event and then proceed with the simulation as before until the first busy cycle has been completed. Note during this busy cycle other rare events will be allowed to occur as specified by the rare event source statistics. As soon as the first busy cycle has been completed (the queue is empty), another rare event is forced to occur. In this way, there are no non-rare event cycles between each rare event cycle. The result of such a simulation would appear statistically similar to the data shown in Table 4.5.1 only with row 1 removed.

For the new variance reduction technique to work we must be able to estimate the average delay with row 1 of Table 4.5.1 not directly available, but assuming some information concerning the non-rare event source is known. In this case, suppose we only know the arrival rate, λ_1, of the non-rare event source and the average delay given only the non-rare event source, i.e., $E\{T|R=0\}$.

#Rare (R=k)	#Occurences	# Total Packets Given R=k $(N|R=k)$	Total Packet Delay (sec) Given R=k $(\hat{T}_T|R=k)$	Packet Delay given R=k $\hat{T}|R=k$ (sec)	Probability R=k $\hat{P}(R=k)$	Biased Probability $\hat{P}_B(R=k)$
0	303403	757791	1132200	1.494	.758	---
1	57	107758	67588500	627.225	.108	.445
2	9	60714	79086600	1302.609	.061	.251
3	5	33167	30941300	932.894	.033	.137
4	1	14612	17726300	1213.133	.015	.060
5	0	----	----	----	----	----
6	1	26067	82111200	3150.006	.026	.108
Total		1000109	2.7858×10^8			

Table 4.5.1
Typical Simulation Results

Before developing the method consider the mechanism that is providing the variance reduction. By forcing a rare event to occur at the end of every busy period we are essentially increasing P(R=k), i.e., biasing the probability that a packet is associated with a busy cycle with more rare events. This is similar to importance sampling, however, in this case we do not know the induced bias a-priori. The necessary information to unbias the results must be obtained from the observed data. Let $\hat{P}_B(R=k)$ be an estimate for the biased probability $P_B(R=k)$ obtained from the rare event simulation. $\hat{P}_B(R=k)$ is obtained from the rare event simulation by

$$\hat{P}_B(R=k) = \frac{N_B|R=k}{N_{B_T}} = \frac{N|R=k}{N_{B_T}}$$

where

$N_B|R=k$ = number of packets associated with R=k in the rare event simulation

N_{B_T} = total number of packets in the rare event simulation

It is important to emphasize that by construction, $(N_B|R=k) = (N|R=k)$, i.e., the number of packets associated with R=k is the same in the biased and unbiased simulations. This observation is also true of the delay $\hat{T}|R=k$. For the example considered here $\hat{P}_B(R=k)$ is also shown in Table 4.5.1. We can find $\hat{P}(R=k)$ given $\hat{P}_B(R=k)$ as follows: we know that

$N_T = N_{B_T} + N|R=0$ = total number of packets which would have been processed in an unbiased simulation

and

$$\hat{P}(R=k) = \frac{N|R=k}{N_T}$$

Combining these results, we obtain the desired probabilities as

$$\hat{P}(R=k) = \hat{P}_B(R=k) \left[\frac{N_{B_T}}{N_T} \right]$$

Note that the bias is constant in this case. Now to apply this technique we use $\hat{P}_B(R=k)$ estimated directly from the biased simulation. However N_T must be inferred from the biased results and the a-priori knowledge. N_T can be found if we can find $N|R=0$. We can estimate $N|R=0$ as follows: knowing that 100 rare events occurred (from Column 1 and 2 of Table 4.5.1 i.e., 1;57 + 2;9 + 3;5 +4;1 + 5;0 + 6;1 = 100) with an average interarrival time of 10,000 sec, the total length of unbiased simulation would be estimated to be 10^6 sec. The amount of this time used in processing rare events, T_R, can be obtained from the rare event only simlation; in this case $T_R = \sim 242{,}000$ (from sum of rows 2 - 7 Column 3 in Table 4.5.1). Thus, approximately 758,000 sec of real time would have been used to process non-rare event cycles. With $\lambda_1 = 1$ packet/-sec, then $N|R=0 \cong 758{,}000$. Note this is very close to the number given by the direct simulation. Further we assume we know $T|R=0$ a-priori and can estimate $(\hat{T}|R=k)$ from the biased simulation so we can find the unconditional delay as

$$\hat{T} = (T|R=0)\ \hat{P}(R=0) + \sum_{k=1}^{\infty} (\hat{T}|R=k)\ \hat{P}(R=k)$$

when

$$\hat{P}(R=0) = 1 - \sum_{k=1}^{\infty} \hat{P}(R=k)$$

Using $\hat{P}(R=k)$ and $T|R=k$ the unconditional average delay can be calculated in this case, it is 278 sec.

To quantify the performance of the new variance reduction technique 10 replications of the brute force and the rare event simulations were obtained. The results are summarized in Table 4.5.2. The estimated delay and confidence intervals* are approximately identical in the two cases, however simulation run time was reduced by a factor of approximately 5.5.

These results indicate that the concepts of regenerative simulation and rare events can be used to develop a variance reduction technique which provides a significant computationally advantage. The assumptions behind the

* Note from these results a single run average of 278 is reasonable.

Brute Force Approach	
CPU* time required	1196 min
Estimated waiting time	321.5
Confidence interval	±68.5
Simulation of rare events only	
CPU* time required	238 min
Estimated waiting time	368.2
Confidence interval	±59.2

Improvement factor in CPU* time required = 5.48

* All simulations were run on a VAX750.

Table 4.5.2
Run Time Comparison

new approach are: 1) a class of "rare" events must exist; 2) these "rare" events must have a significant impact on the system performance; 3) the average delay given no rare events must be known (this information can be analytically obtained or through a separate simulation; and 4) because in this case a rare event was forced wherever the queue was empty, the effect of a rare event occurring when short messages were in the queue was ignored. Clearly, in the case considerd here this is an excellent assumption.

Further work is needed to extend and quantify this approach. For example, it is clear from Table 4.5.1 that the busy cycles containing several rare events are more significant (i.e., contribute more to the average delay) compared to those containing a smaller number. Thus, techniques could be needed to increase the bias of the large k cycles relative to the others. This might be done by forcing another rare event to occur when the queue is in some given state (other than empty). The technique needs to be extended to networks of queues. The theoretical basis for the method also needs to be fully developed.

4.5.3.5 Usefulness of Variance Reduction Techniques for PCNSAS

Variance reduction techniques can be effective for reducing the computational requirements for network simulation. As noted by Heidelberger and Lavenberg [19], while variance reduction techniques are an attractive possibility for cutting down on simulation run lengths and have been much studied in the literature, they have been rarely successful in real applications. It is further concluded in [19] that effective generally applicable variance reduction techniques need to be developed. This observation is especially true for PCNSAS, the variance reduction technique must be generally applicable and worth the effort.

Of the methods investigated here, the new variance reduction technique based on rare events seems to hold the most promise, but is not generally applicable, i.e., the system must have low and high frequency events. However, for the classes of networks to be studied, e.g., packet radio networks in a dynamic environment, the rare event variance reduction technique should provide significant computational advantages.

The variance reduction techniques based on regenerative simulation with control variables have been effective for the cases studied, however, the difficulty in selecting the appropriate control variable reduces the current attractiveness of these methods.

4.6 Conclusions and Recommendations

Obtaining statistically significant results from simulation models of data communications networks is a non-trivial task. A variety of techniques have evolved over the years to aid in obtaining and interpreting simulation results. These methods must be incorporated into PCNSAS. Specifically, it is recommended that regenerative techniques be incorporated into the simulation models and post-processor. Regenerative methods provide a straight forward technique for obtaining confidence intervals and are the basis of many of the variance reduction techniques.

The major problem that must be overcome before PCNSAS can become a useful tool is computational efficiency. That is, we must be able to model and simulate large PCN with realistic traffic and in realistic environments and obtain reasonable confidence concerning the performance measures with finite computer resources. There are six areas identified below in which further research on computational methods for data communications network simulation is needed.

4.6.1 Develop an Aggregate Background Traffic Model Networks

Partitioning nodes into primary and background stations, then simulating the primary nodes in detail while modeling all the background stations as a single node proved effective for modeling CSMA/CD networks. Similar techniques for packet networks could also prove quite effective. Research is needed to develop suitable background models and then evaluate their effectiveness. Such model must not only account for the media access protocol as is the previous study [37] but must describe the effects of higher levels protocols on the background traffic. A significant effort will be required to develop such background traffic models but the savings in computational resources could be significant.

4.6.2 Investigate the Potential of Combined Hybrid Analytic/Simulations Models for G/G/1 Queue

A combined analytic/simulation model was proposed in Section 4.5.2.5. This technique was based on the analytic expression for the average waiting time in an G/G/1 queue where some terms in the expression were analytically evaluated while others were estimated using simulation. This technique is simple and easy to implement using regenerative methods. However, its effectiveness is unknown. Resarch is needed to investigate its effectiveness. Further work is needed to evaluate its usefulness as applied to networks of queues. A moderate effort will be needed to conduct the required studies.

4.6.3 Investigate the Potential of General Hybrid Models

As previously noted, hybrid models can be very effective in reducing the computational requirements for specific networks. However, no generally applicable hybrid techniques exist today. A long term research effort is needed to study and develop a modular approach to hybrid simulation. Such a modular approach must allow for the interchange of different modules, representing for example different media access protocols. Further, the approach must allow for the easy construction of new modules. A significant effort will be required to investigate possible approaches.

4.6.4 Investigate Perturbation Analysis Techniques

The progress in research on perturbation analysis should be followed. As the techniques and its limitations become better understood, its application to PCNSAS should be considered. The benefit for developing optimization methods for PCNSAS could be significant through the applicaton of perturbation analysis concepts.

4.6.5 Extend the Rare Event Based Variance Reduction Technique to Networks and Evaluate its Performance

The rare event based variance reduction technique described here shows promise for reducing the computational requirements for systems which have low and high frequency events. Research is needed to further develop a theoretical basis for the technique. The expected performance improvement for this method must be quantified. Extension of the technique to networks of queues is needed.

4.6.6 Investigate Regenerative Methods with Control Variables

Several studies have been reviewed here (see Section 4.5.3.2) which use the regenerative method with control variables to obtain a variance reduction. Research is needed within the context of data communications networks to evaluate the effectivness of these techniques. Specifically, control variables which are applicable in a network environment should be sought. It is expected that only modest variance reductions can be obtained using control variables in a network environment.

4.6.7 Summary of Future Research Efforts

Four combined analytic/simulation and two variance reduction techniques have been identified for further study. The investigation of general hybrid background traffic models and perturbation analysis have the most potential for providing significant computational efficiency. However, these two methods also require the most research effort. The rare event based variance reduction techniques and the G/G/1 hybrid model offer the most near term potential for obtaining some computational efficiencies with modest required research effort. The study of control variables for network simulation will require a significant effort with only modest improved efficiencies expected. Each of the techniques evaluated here will impose different constraints on the structure of PCNSAS. A problem for Phase II is to address how PCNSAS can be structured so that it has the flexibility to accommodate different variance reduction techniques as they mature.

5.0 CONCLUSIONS

Simulations play an important role in the analysis and design of communication links and networks, and there is a need for developing simulation based CAD tools for networks. In this report we have presented the results of a preliminary effort to develop a state-of-the-art network simulator. Our work has shown the feasibility of a modular and computationally efficient approach that can be used for developing a general purpose network simulator that can be used in a variety of applications. A prototype simulator has been built to show the feasibility of our approach and the use of simulations as an analysis tool. Several research problems have been identified and approaches for solving these problems have been presented.

Results of this preliminary (Phase I) effort have clearly demonstrated the feasibility of our approach to developing an integrated CAD tool for networks. Additional research and development work needs to be completed in Phase II. Successful completion of Phase II will result in a network CAD tool that will have many applications within DoD, its components and contractors, and in the civilian sector.

6.0 REFERENCES

1. Special Issue on Computer-Aided Modeling, Analysis, and Design of Communication Systems, IEEE Journal on Selected Topics in Communications, P. Balaban, K. Sam Shanmugan, and B. W. Stuck, Editors, January, 1984

2. S. Shoemaker, Computer Networks and Simulation, North-Holland, 1978.

3. S. Shoemaker, Computer Networks and Simulation II, North-Holland, 1982.

4. J. F. Hayes, Modeling and Analysis of Computer Communication Networks, Plenum Publishers, 1984.

5. A. B. Prisker, Introduction to Simulation and SLAM II, Halsted Press, 1984.

6. C. H. Sauer and E. A. MacNair, Simulation of Computer Communications Systems, Prentice-Hall, Inc., Englewood, N.J., 1983.

7. A. M. Law and C. S. Larmey, "An Introduction to Simulation Using SIMSCRIPT 11.5," C.A.C.I. Los Angeles, California, 1984.

8. T. J. Schriber, Simulation Using GPSS, Wiley, New York 1974.

9. Catalog of Simulation Software, Simulation, October, 1985, pp. 196-209.

10. K. Shanmugan et al., " A Block Oriented Systems Simulator (BOSS)", Technical Report, Telecommunications Laboratory, University of Kansas, December,1985.

11. Stefik, M., and Bobrow, D. G., "Knowledge Programming in LOOPS," AI Magazine, Vol. 4, No. 3, Fall 1983.

12. P. J. Kuehn, "Modeling and Analysis of Computer Networks, Decomposition Techniques, Transient Analysis and Protocol Implication," ICC '84, May 1984, Paper 41.6.

13. J. G. Shanthikumar and R. G. Sargent, "A Unifying View of Hybrid Simulation/Analytic Models and Modeling," Oper. Res., Volume 31, No. 6, November, December 1983.

14. J. P. C. Kleijnen, Statistical Techniques in Simulation, Part I, Marcel Dekker, Inc., New York, 1974.

15. Kleinrock, L., Queueing Systems, Volume I and II, John Wiley, New York, 1975, 1976.

16. Lavenberg, S. S., (ed.) Computer Performance Modeling Handbook, Academic Press, New York, 1983.

17. E. D. Lazowska, et al., Quantative System Performance Computer System Analysis Using Queueing Network Models, Prentice Hall, 1984.

18. R. Reiser, "Performance Evaluation of Data Communications Systems," Proceedings of IEEE Vol. 70, No. 2, Fed. 1982.

19. P. Heidelberger and S. Lavenberg, "Computer Performance Evaluation Methodology", IEEE Trans. on Computers, Vol. C-33, No. 12, 1984.

20. M. Ilyas and H. Mouftah, "Performance Evaluation of Computer Communications Networks," IEEE Comm. Mag. Vol. 23, No. 4, April 2, 1985.

21. E. M. Auppeal, "Merits Evoluation Statistically Speaking," IEEE Transactions on Computers, Vol. C-32, No. 10 pp. 881-902, October 1983.

22. W. T. Marshall and S. O. Morgan," Traffic Measurements on the Murray Hill DATAKIT Network," TM 11273-8407-01, August 1984.

23. M. Schwartz, Computer-Communications Network Design and Analysis, Prentice-Hall 1977.

24. G. S. Fishman, Concepts and Methods in Discrete Event Digital Simulation, John Wiley, 1973.

25. G. S. Fishman, Principles of Discrete Event Simulation, John Wiley, 1978.

26. L. W. Schruber, "Detecting Initialization Bias in Simulation Output," Oper. Res. Vol. 30, pp. 569-590, 1982.

27. P. Heidelberger, "Variance Reduction Techniques for the Simulation of Markov Processes, I, Multiple Estimates," IBM J. Res. Develop. Vol. 24, No. 5, September, 1980.

28. W. Whitt, "Embedded Renewal Processes in the GI/G/S Queue," J. Appl. Prob. Vol. 9, pp 650-658, 1982.

29. M. A. Crane and A. J. Lemoine, An Introduction to the Regenerative Method for Simulation Analysis, Springer-Verlag, New York, 1977.

30. M. A. Crane and D. L. Iglehart, "Simulation Stable Stochastic Systems, I: General Multiserver Queues," J. Association for Computing Machinery, Vol. 21, No. 1, January, 1974.

31. M. A. Crane and D. L. Iglehart, "Simulating Stable Stochastic Systems III, Regenerative Processes and Discrete Event Simulation," Oper. Res. Vol. 23, No. 1, January-February, 1975.

32. J. R. Wilson and A. B. Prisker, "A Survey of Research on the Simulation Start-Up Problem," Simulation, Vol. 31, pp. 55-58, 1978.

33. L. Schruber et al., "Optimal Tests for Initialization Bias in Simulation Output," Oper. Res. Vol. 31, No. 6, November-December, 1983.

34. A. M. Law, "Statistical Analysis of Simulation Output Data," Oper. Res. Vol 31, No. 6, November-December, 1983.

35. S. Ehreufeld and S. Bertuvia, "The Efficiency of Statistical Simulation Procedures," Technomatics, Vol. 4, No. 2, May, 1962.

36. J. W. Wong, J. A. B. Moura, J. A. Field, "Hierarchical Modeling of Local Area Computer Networks," Globecom '80, 1980.

37. P. J. O'Reilly and J. L. Hammond, Jr., "An Efficient Simulation Technique for Performance Studies of CSMA/CD Local Networks," IEEE Journal on Selected Areas in Communications, Vol. SAC-2, No. 1, January, 1984.

38. W. A. Moy, "Variance Reduction," Chapter 10 in Computer Simulation Experiments with Models of Economic Systems, T. J. Naylor Ed. Wiley, 1971.

39. G. S. Fishman, "Accelerated Accuracy in the Simulation of Markov Chains." Oper. Res. Vol 31, No. 3, June, 1983.

40. K. S. Shanmugan and P. Balaban, "A Modified Monte-Carlo Simulation Technique for Evaluation of Error Rate in Digital Communication Systems," IEEE Trans. Comm. Vol. COM-28, November, 1978.

41. R. L. Mitchell, "Importance Sampling Applied to Simulation of False Alarm Statistics," IEEE Trans. on Aerospace Electronics, Vol. AES-17, No. 4, January, 1981.

42. M. C. Jeruchim, "Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communications Systems," IEEE Journal on Selected Areas in Communications, Vol. SAC-2, No. 1, 1984.

43. P. A. Lewis and D. L. Iglehart, "Regenerative Simulation with Internal Controls," J. ACM Vol. 26, No. 2, April, 1974.

44. S. S. Lavenberg, T. L. Moeller and P. D. Welch, "Statistical Results on Control Variables with Application to Queueing Network Simulation," Oper. Research Vol. 30, No. 1, January-February, 1982.

45. R. Y. Rubinstein and R., Marcus, "Efficiency of Multivariate Control Variates in Monte-Carlo Simulation," Oper. Research, Vol. 33, No. 3, May-June, 1985.

46. R. V. Hogg and A. T. Craig, Introduction to Mathematical Statistics, MacMillan Co., 1965.

47. S. Tolopka and H. Schwetman, "Mix-Dependent Job Scheduling, An Application of Hybrid Simulation," National Computer Conference, 1979.

48. H. D. Schwetman, "Hybrid Simulation of Computer Systems," Comm. of ACM, Vol. 21, No. 9, Sept. 1978.

49. W. W. Chiu and W. M. Chow, "A Performance Model of MVS," IBM J. Res. Develop, Vol. 17, No. 4, 1978.

50. D. L. Minh and R. M. Suri, "Simulating the GI/G/1 Queue in Heavy Traffic," Oper. Research, Vol. 31, No. 5, September-October, 1983.

51. A. M. Law, "Efficient Estimators for Simulated Queueing Systems," Management Science, Vol. 22, No. 1, September, 1975.

52. G. Carter and E. J. Ignall, "Virtual Measures, A Variance Reduction Technique for Simulation," Management Science, Vol. 21, No. 6, February, 1975.

53. Y. C. Ho and X. Cao, "Perturbation Analysis and Optimization of Queueing Networks," J. Optimization Theory Applications, Vol. 40, No. 4, August 1983, pp. 554-582.

54. C. G. Cassandras and S. G. Strickland, "Perturbation Analysis Techniques for Communications Networks," GlobeCom 1985, pp. 13.5.1-13.5.5, December 1985.

55. A. M. Law and W. D. Kelton, Simulation Modeling and Analysis, McGraw-Hill Book Co., 1982.

END
1-87
DTIC